

CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen IX-Fascículo 102

Director:	José Mas Godayol
Director editorial:	Gerardo Romero
Jefe de redacción:	Pablo Parra
Coordinación editorial:	Jaime Mardones
	Francisco Martín
Asesor técnico:	Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco

Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redaccion y administración:
Aribau, 185, 1.°, 08021 Barcelona
Tel. (93) 209 80 22 - Télex: 93392 EPPA

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-181-X (tomo 9)
84-85822-82-X (obra completa)
Depósito Legal: B. 52-84

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 318512
Impreso en España-Printed in Spain-Diciembre 1985

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 19 425 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 429 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S. A. (Aribau, 185, 1.°, 08021 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

No se efectúan envíos contra reembolso.



Los medios de producción

Cortesía del doctor Johannes Heidenhain, RFA

Veamos cómo las máquinas controladas por ordenador fabrican productos acabados

En el pasado, los ingenieros leían los dibujos y preparaban a mano sus máquinas-herramienta. Las máquinas de control numérico (CN) introdujeron un proceso más rápido y de mayor precisión: la máquina-herramienta se prepararía a sí misma de acuerdo a instrucciones recibidas en virtud de un lenguaje de instrucciones. Los diseñadores producirían dibujos técnicos incorporando todas las mediciones adecuadas. Los ingenieros de producción traducirían luego los dibujos a programas de CN. La máquina-herramienta tendría, entonces, toda la información necesaria para mecanizar la pieza.

Los programadores de CN disponen de un conjunto de instrucciones estandarizado en la industria para definir los tipos de movimientos que puede realizar una máquina. El programador define un movimiento y luego añade las coordenadas precisas. Asimismo, necesita definir la velocidad a la cual la máquina está girando o rotando. Las tareas complejas a menudo se efectúan por etapas, avanzando a través de una forma estilizada hasta la pieza ya acabada. Ahora este sistema está siendo superado gradualmente por las máquinas controladas numéricamente por ordenador (CNC: *computer numerically controlled machines*), en las que la cinta de papel que siempre ha alimentado a las máquinas de CN se sustituye por una cinta magnética o por discos flexibles. En los más elaborados de estos sistemas, las empresas están considerando la posibilidad de pasar directamente del CAD a programas de CN generados automáticamente.

No todos los sistemas de CAD son suficientemente precisos para hacer esto, pero, en caso de que lo sean, se puede utilizar de forma cabal la base de datos CAD central en todas las etapas de diseño y producción. El ingeniero de producción, quien tiene una biblioteca de máquinas-herramienta a su disposición, puede acceder a la base de datos CAD central, que posee una descripción geométrica completa de la pieza. Elige la máquina-herramienta que necesita para mecanizar la pieza y, a medida que escribe el programa, puede ver cómo la herramienta se va moviendo a lo largo del contorno del dibujo. De esta forma se pueden evitar errores simples pero costosos, como dirigir la herramienta hacia adelante en lugar de hacia atrás.

El uso de la misma base de datos también acelera el proceso, porque cualquier cambio que introduzca el diseñador en el curso del desarrollo se transmite de forma instantánea al ingeniero. Al diseñador se le pueden indicar directamente los fallos de diseño que podrían hacer imposible la fabricación de una pieza.

El control de procesos modernos es esencial para el funcionamiento continuo de casi todas las industrias, y, por lo general, se lleva a cabo mediante conjuntos de microprocesadores. Un sistema de control de procesos consta de cinco componentes cruciales:

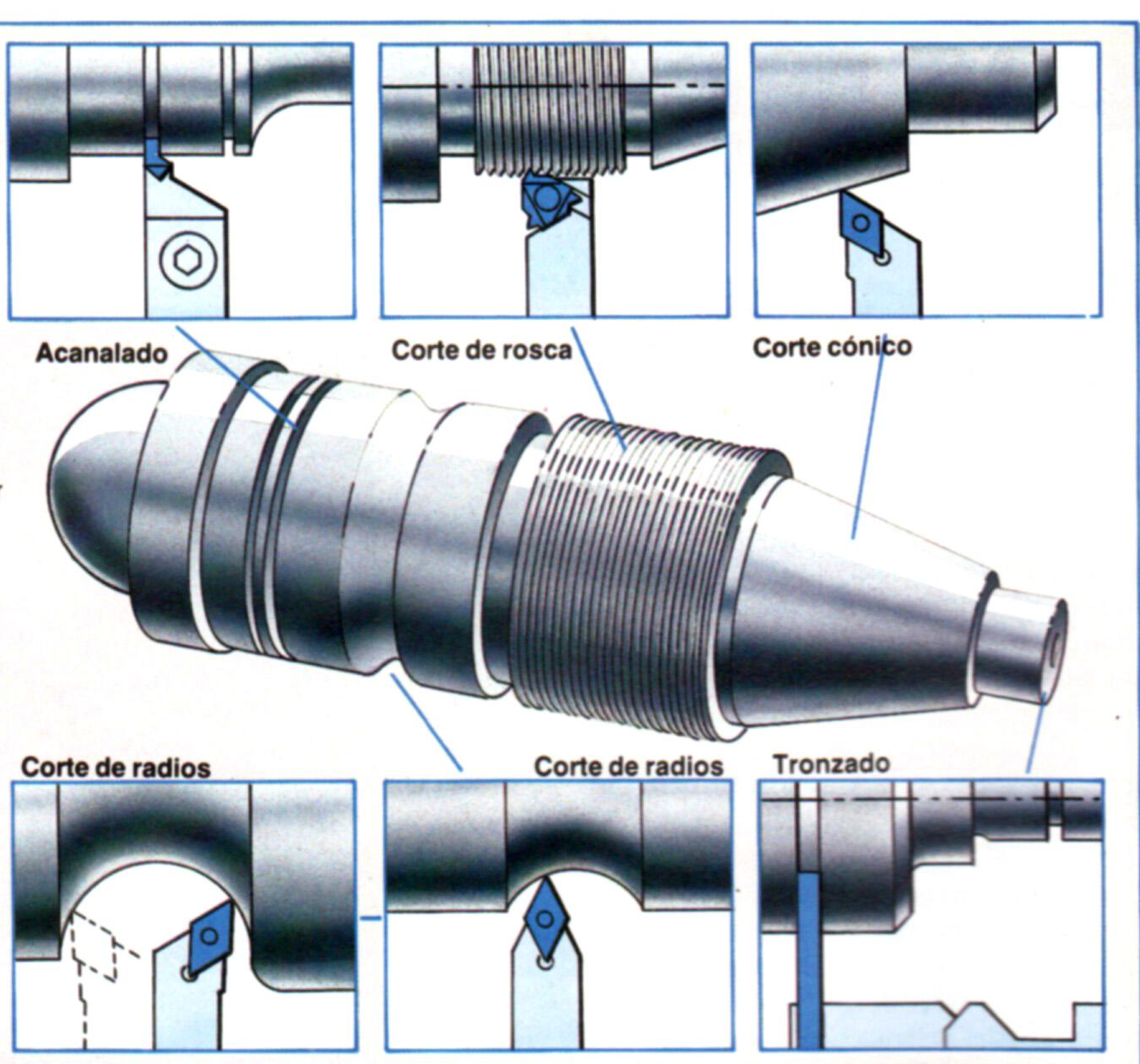
- Un *sensor* para medir la característica determinada, como temperatura, caudal o presión.
- Un *actuador* capaz de iniciar una acción en respuesta a una señal.
- También ha de haber *una válvula de control*, o equivalente. Ésta será activada por el actuador para efectuar el proceso.
- Un *controlador* determina la acción a tomar, basado en la información recibida desde el sensor.

Trabajando

Los progresos en el procesamiento del lenguaje han permitido que se construyan interfaces para máquinas de CNC capaces de aceptar especificaciones en lenguaje corriente. Las primeras máquinas de CN no eran tan eficientes, puesto que necesitaban que un personal altamente entrenado les introdujese una complicada serie de códigos. En el futuro se podrían producir máquinas "inteligentes" que realizaran tareas diferentes en distintos lugares de la fábrica

Herramientas de precisión

Las máquinas de CNC pueden llevar a cabo varias operaciones diferentes sobre el mismo componente mediante la selección de distintas herramientas de corte, tal como le indica el programa maestro, introducido ya sea por el ingeniero o bien directamente desde el sistema de ordenador central. El control por microprocesador permite que una misma máquina domine todas las principales técnicas de mecanizado de un torno convencional con una supervisión mínima



Control de procesos

El verdadero trabajo de un sistema de control de procesos lo efectúan los sensores de proceso, que leen, miden e interpretan los datos. La mayoría de ellos contienen microprocesadores.

Medición de temperatura

- *Pila termoeléctrica:* Es la forma más económica y corriente de medir la temperatura. Una pila térmica usa dos metales con distintos coeficientes de dilatación por el calor. Entre ellos se genera una tensión, que es la base de una señal analógica.
- *Termómetro de resistencia:* Se basa en el principio de que el calor aumenta la resistencia eléctrica de los cables conductores. La corriente pasa por un hilo calentado y uno sin calentar. Se mide la diferencia de tensión y se calcula el cambio en la resistencia.
- *Pirómetro de radiación:* Mide el calor radiado y se utiliza para temperaturas más elevadas que los dos dispositivos anteriores. El pirómetro concentra el calor radiado en dispositivos que esencialmente son pilas termoeléctricas y mide la temperatura desde dicho punto de concentración.

Medición de flujo

- *Medidores de presión diferencial:* Calculan la velocidad de flujo midiendo la presión en diferentes puntos a ambos lados de un estrechamiento.

Medición de presión

- *Tubo Bourdon:* Utiliza un tubo de metal en forma de C, en espiral o en hélice. Cuando se aplica presión en un extremo, el tubo intenta enderezarse. La presión se mide por el desplazamiento del extremo libre.
- *Manómetro:* Tubo en forma de U con un brazo más ancho. En éste se monta un flotador. La presión en el brazo más largo y más estrecho varía el nivel de la superficie del líquido en el otro.
- *Diafragma:* Un acoplamiento mecánico mide el movimiento ejercido en un delgado diafragma de goma. En el fuelle hay un sensor más complejo, que se transfiere a la parte inferior y se mide.

Medición de nivel

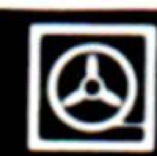
- *Dispositivos operados con flotador:* Se miden los

- Para hacer esto, el *transmisor* debe tomar la señal del sensor y pasársela al controlador de una forma "comprensible".

La mayoría de estos dispositivos son analógicos, enviando su información en forma de tensiones. Por este motivo, el sistema contiene un convertidor de analógico a digital que cambia la señal a una forma que le resulte aceptable al ordenador. Los sistemas de control de procesos dependen de tales dispositivos situados en puntos clave de la planta de proceso y casi todos ellos ejecutan el mismo programa una y otra vez. Podrían, por ejemplo, tener que ejecutar un programa una vez por segundo para comprobar el caudal de un flujo en una tubería. Si su velocidad superara los límites establecidos, entonces se habrían de activar las válvulas de control.

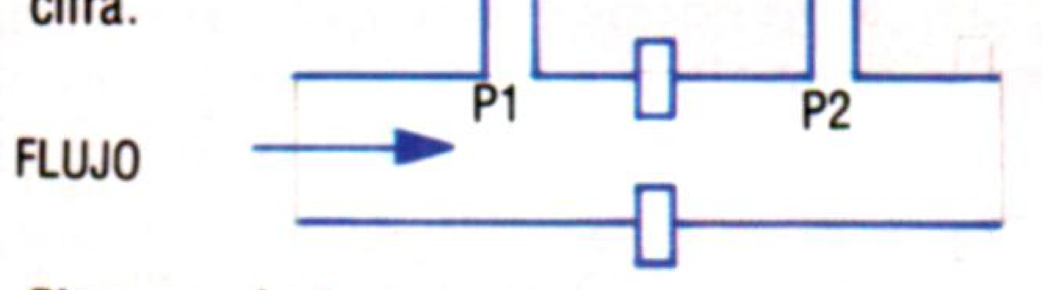
Si bien los sistemas de control de procesos son tan diversos como los procesos que controlan, se pueden dividir en dos tipos principales: de control continuo y de control secuencial. Ambos no son mutuamente excluyentes y, de hecho, se pueden incluir en el mismo sistema.

El control continuo ve cómo la materia prima se suministra a un proceso y atraviesa el mismo sin

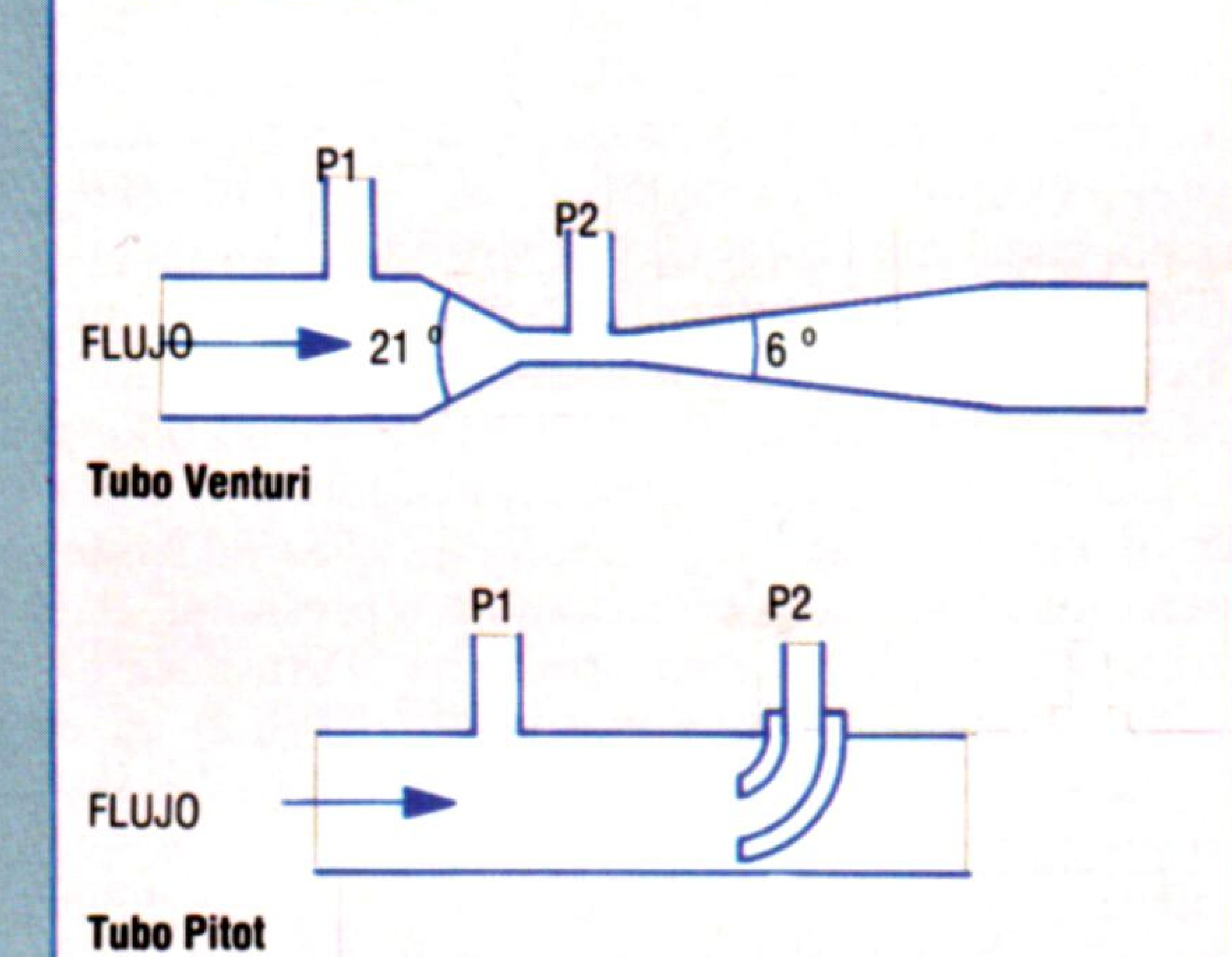


Flujo rápido

El flujo de líquidos y gases se puede medir y transmitir a un ordenador anfitrión monitorizando la presión diferencial. En cada caso la presión se mide en dos puntos, P1 y P2. La presión diferencial es igual a P1-P2, y el flujo en el tubo es directamente proporcional a esta cifra.



Placa con aberturas



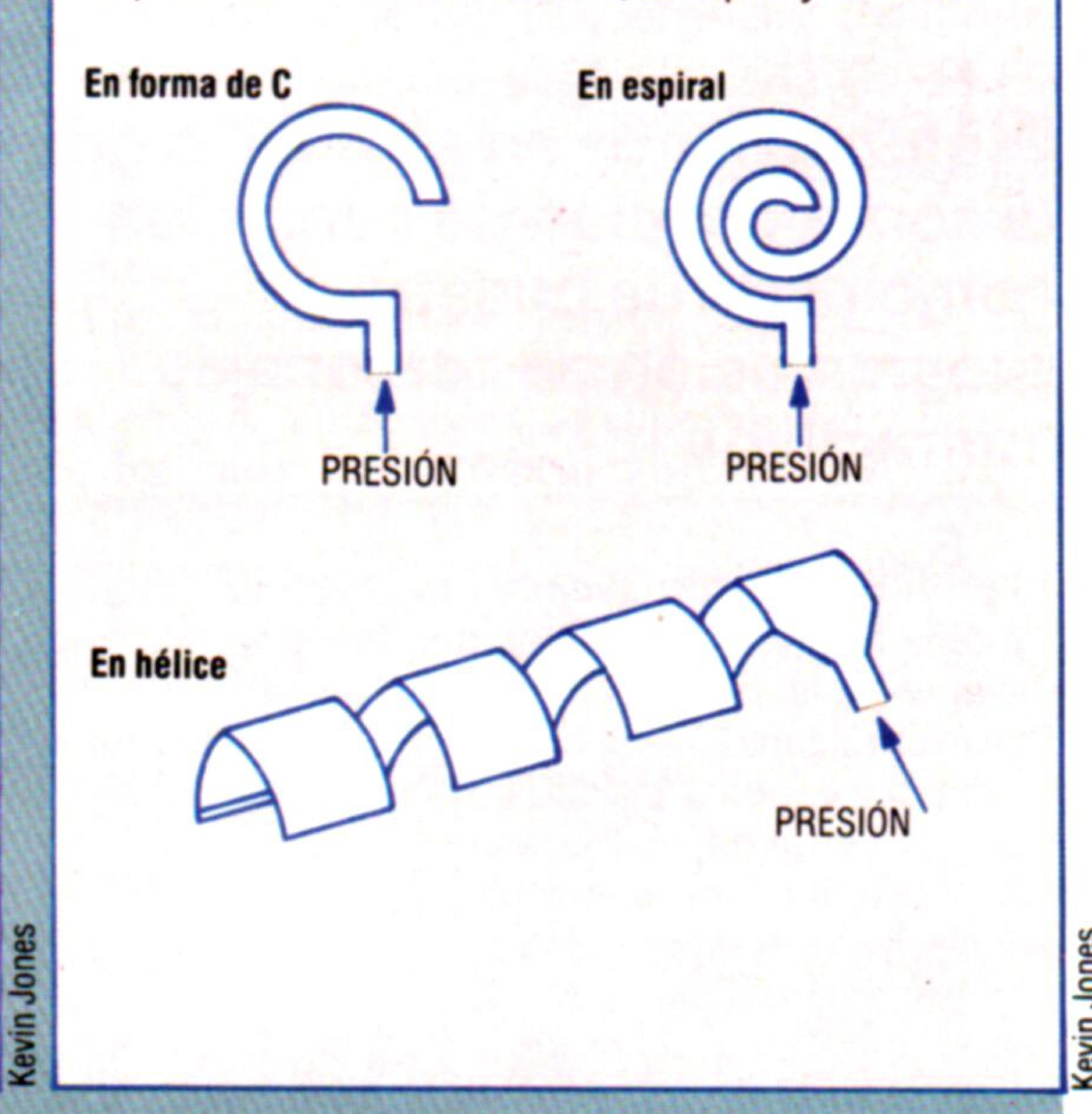
Tubo Venturi

Tubo Pitot

Kevin Jones

Bajo presión

El tubo Bourdon consiste en un tubo metálico que, cuando se le aplica presión al dispositivo, tiende a enderezarse. Entonces se mide el movimiento del tubo para obtener una indicación de presión y el resultado se le transmite al ordenador para su análisis. Los tubos Bourdon se presentan en tres formas: en C, en espiral y en hélice



Kevin Jones

cambios de nivel a medida que el flotador sube o baja.

- *Dispositivos de presión:* La presión en la parte inferior de un depósito mide la cantidad y nivel del líquido contenido.
- *Radiación:* Una célula a uno de los lados del recipiente calcula cuánta radiación gamma está pasando al otro lado desde una fuente. La radiación sólo pasará allí donde el líquido no obstruya su camino y, de este modo, se puede medir el nivel.
- *Sensor:* Calcula la profundidad midiendo el tiempo de respuesta ante una señal ultrasónica.

Medición de peso

- *Células de carga de resistencia eléctrica:* Se basan en que los conductores cambian su resistencia al someterse a tensión mecánica.
- *Células semiconductoras:* Materiales semiconductores, como el silicio, producen tensión bajo presión y pueden usarse como medida.

Análisis químicos

- *Infrarrojos:* Esta forma de análisis se utiliza para medir gases. Se pasan muestras del gas a través de una fuente de infrarrojos hasta un detector de infrarrojos. La cantidad de infrarrojo bloqueado en el camino es una medida del gas presente.
- *Analizadores de conductibilidad térmica:* Los cambios en un gas alteran su capacidad para conducir calor. Se pasa el gas a través de un elemento calefactor a una velocidad constante y se observa si cambia la conducción de calor.

Velocidad

- *Generador taquimétrico:* Mide una velocidad de rotación. El generador se fija a un eje rotativo y, a medida que éste gira, genera corriente. Puesto que la corriente es proporcional a la velocidad, se puede medir esta última.

En los sistemas de control de procesos, la salida de estos sensores, así como la de otros muchos, se aplica a un controlador de tres condiciones. Éste toma el valor medido por los sensores y lo compara con un valor establecido por el operador, que es el nivel al cual se supone debe iniciarse la acción. Si el valor se halla por encima o por debajo del punto establecido, puede activar la válvula y cambiar el proceso en la forma requerida

detenerse. Es necesario el control de la velocidad de flujo a lo largo de todo el camino. El sistema ha de ser capaz de responder a pequeños cambios con el objeto de mantener las condiciones óptimas. En el control secuencial, el proceso se controla por tandas. La materia prima ha de someterse a varios cambios físicos o químicos, de modo que el control se limita a iniciar una serie de acciones en el momento correcto y en las condiciones correctas. Asimismo, debe controlar todas las diferentes etapas para asegurarse de que cada una obtenga el mismo tratamiento.

La gran ventaja del uso de ordenadores en el control de procesos es que permiten hacer muchas cosas al mismo tiempo. Los procesos particulares se pueden controlar individualmente mediante componentes estándares que se pueden intercambiar en caso de una avería. Por sobre todo, el control de procesos informatizado permite que un solo supervisor de planta controle muchas operaciones diferentes desde un único terminal. En vez de tener que leer centenares de diales y medidores, puede seguir la pista de lo que está sucediendo observando los "diagramas simulados" (representaciones estilizadas de la planta y sus procesos) que se visualizan en su VDU.

Árbol de decisión

Profundizamos en el empleo de la comprobación de condición como parte de nuestra programación de personajes interactivos

El problema que estudiamos en el capítulo anterior suponía la comprobación de una secuencia de condiciones y la ulterior impresión de un mensaje. Pero supongamos, por ejemplo, que no queremos responder a todas las condiciones, sino sólo a algunas. Esto sería difícil utilizando solamente ON...GOTO, puesto que habríamos de hacer juegos malabares con las distintas condiciones con el fin de terminar con una secuencia de valores para que la sentencia funcionara del modo adecuado. Nuevamente, los árboles de decisión son la respuesta.

Volviendo a nuestro ejemplo anterior de las cuatro condiciones (indicando humano, animal, macho o hembra), el siguiente diagrama muestra un árbol diseñado para comprobar las distintas condiciones y rechazar todas las entradas del usuario, a excepción de aquellas que confirmen que el usuario es un ser humano macho o hembra.

Hombres y mujeres solamente
El descenso a través del árbol que vemos aquí llevará al usuario a un nudo terminal "invalidado", a menos que las preguntas hayan sido respondidas por un humano macho o hembra. Si bien el diseño es semejante al del ejemplo de nuestro anterior capítulo, este árbol posee nudos terminales en niveles distintos, no sólo en el nivel 5. Los nudos de elección se han numerado en color azul, los nudos terminales en rojo y negro. Los números rojos indican resultados válidos (es decir, humanos machos o hembras)

Diferencias de género

Nivel 1
Nivel 2
Nivel 3
Nivel 4

Humano · Animal · Hembra · Macho · Invalidado

Puede observarse que entre este árbol y el anterior hay ciertas semejanzas. Sigue habiendo cuatro niveles y (remitiéndonos al listado del capítulo anterior) cada nivel comprueba una condición retenida en el elemento de la matriz c correspondiente a ese nivel. Sin embargo, los números de los nudos no guardan la misma relación que antes. Ello se debe a que un nudo terminal no tiene que hallarse necesariamente en la parte inferior del árbol, sino que puede estar en uno de los niveles anteriores, como es el caso del nudo 9, por ejemplo.

Para integrar este nuevo árbol en nuestro programa, primero hemos de numerar los nudos; en este caso, del 1 al 6. Esto nos proporcionará una forma de comprobar, cuando recorramos el árbol, si hemos llegado a un nudo terminal, puesto que cada uno tendrá un número mayor que seis.

En esta etapa podemos simplificar aún más las cosas asegurándonos de que el siguiente conjunto de números que numeremos sean aquellos nudos terminales que requieran que tomemos alguna acción específica: en este caso, los números siete y ocho, cada uno de los cuales representa una de las condiciones que estamos buscando. Por último, numeramos los nudos terminales que no nos interesan.

Las razones que nos mueven a seguir este orden de numeración se harán evidentes más adelante; pero mientras tanto necesitamos representar este árbol dentro de nuestro programa. Para hacerlo, utilizaremos una nueva matriz t(6,1) (t(6,2) en el Spectrum) que retendrá la información de los distintos nudos de elección y de los nudos a los que saltan según de qué condición se trate. La tabla muestra los distintos valores para la matriz t.

Entre el siguiente listado, construido entre las líneas 10 y 80 del listado que ofrecimos en el último capítulo, y las rutinas de bajo nivel de nuestro programa *Dog and Bucket*. Antes ya imprimimos los complementos para las rutinas de bajo nivel en el Spectrum, el Commodore 64 y el BBC Micro. Observe que hemos incluido algunas líneas nuevas entre las líneas 20 y 30, y añadido un número de

Tabla del árbol

La matriz t(6,1) —t(6,2) en el Spectrum, que no siempre permite elementos cero en las matrices— retiene los datos necesarios para nuestra estructura arborescente. Si al llegar al nudo 4, p. ej., encontramos que la condición retenida en c(4) es falsa, bifurcamos al número de nudo retenido en el miembro t(4,0) de la matriz. Si es verdadera, la bifurcación se efectúa hasta el número retenido en t(4,1). Los usuarios del Spectrum bifurcarán utilizando t(4,1) y t(4,2), respectivamente

Número de nudo	Falso	Verdadero
1	2	3
2	9	10
3	4	11
4	5	6
5	12	7
6	8	13

línea 500 adicional. Estas nuevas líneas leen en la matriz t los datos correspondientes. Asimismo, hemos añadido una sentencia DIM al final de la línea 10, DIM t(6,1).

```
10 h$="humano  ":a$="animal  ":m$="macho  ":f$=hembra
   ":q$="Eres un ":DIM c(4),t(6,1)
20 GOSUB 4050:REM limpiar la pantalla
22 REM preparar matriz árbol
24 c=6:REM esta es la cantidad de nudos de elección
26 FOR x=1 TO c: READ t(x,1),t(x,0):NEXT x
30 REM preparar las cuatro variables
```

```
  40 PRINT q$;h$;:INPUT i$:c(1)=ABS(i$="s" OR i$="S")
  50 PRINT q$;a$;:INPUT i$:c(2)=ABS(i$="s" OR i$="S")
  60 PRINT q$;f$;:INPUT i$:c(3)=ABS(i$="s" OR i$="S")
  70 PRINT q$;m$;:INPUT i$:c(4)=ABS(i$="s" OR i$="S")
  80 PRINT
 500 DATA 3,2,10,9,11,4,6,5,7,12,13,8
4000 REM
4010 REM subrutinas del sistema de bajo nivel
4020 REM
4030 REM limpiar la pantalla
4040 REM
4050 CLS:RETURN
4060 REM
4070 REM beep
4080 REM
4090 PRINT CHR$(7);:RETURN
4100 REM
4110 REM tomar un caracter del teclado
4120 REM
4130 i$=INKEY$:IF i$="" GOTO 4130
4140 RETURN
```

Ahora añada las siguientes líneas, que recorren el árbol e imprimen un mensaje adecuado:

```
 90 GOSUB 210:REM llamar rutina clasificacion arbol
100 PRINT "Pulsa una tecla para continuar..."
110 GOSUB 4130:REM toma un caracter
120 GOTO 40
200 REM clasifica arbol
210 n=1: k=1:REM empezar en nudo 1,nivel 1
220 n=t(n,c(k)):k=k+1:IF n<=c THEN 220
230 ON n-c GOTO 250,260
240 PRINT "Entrada invalidada...Por favor vuelve a probar":RETURN
250 PRINT "Eres un hombre...":RETURN
260 PRINT "Eres una mujer...":RETURN
```

Si ejecuta este programa, verá que rechaza todas las entradas excepto aquellas efectuadas por "auténticos" hombres o mujeres. El árbol se recorre de forma similar a nuestro primer ejemplo, pero con las siguientes modificaciones. Primero, no sabemos necesariamente cuántos niveles descenderemos antes de encontrar un nudo terminal, de modo que no podemos utilizar un bucle simple FOR n TO numero de niveles. En cambio, inicializamos una nueva variable, k, en la línea 140, para llevar el registro del nivel en donde nos hallamos.

Habiendo establecido n para retener el número del nudo inicial (1), descendemos entonces por el árbol en la línea 150, utilizando la fórmula n=t(n,c(k)) para tomar el siguiente número de nudo de la matriz t. Luego descendemos un nivel (k=k+1) y, si el número de nudo actual es menor o igual que el número de nudos de elección (n=c), sabemos que aún no hemos llegado a un nudo terminal, de modo que saltamos hacia atrás hasta el comienzo de la línea y repetimos el proceso.

Ahora puede ver por qué los números están numerados como describíamos más arriba. La división de los nudos terminales en dos categorías (aquellas que representan las dos condiciones finales [macho y hembra humanos] que queríamos captar seguidas por aquellas que queríamos rechazar) nos permite utilizar una sentencia ON...GOTO para imprimir los diferentes mensajes. Como ya hemos señalado, el principal inconveniente del empleo de ON...GOTO es que requiere una secuencia de valores que a menudo resulta difícil acomodar, pero el árbol nos permite hacerlo sin ninguna dificultad.

Pluralidad de condiciones

Hasta ahora nos hemos concentrado en estructuras que sólo tienen una cosa en común: todas las condiciones comprobadas en cada nivel de un árbol han sido las mismas. Esto ha simplificado las cosas, porque no hemos tenido que averiguar qué condición comprobar en cada nudo. Si, por ejemplo, nos encontráramos en un nudo en el nivel 3, sabríamos que la condición a comprobar está retenida en c(3). Lamentablemente, las cosas no siempre son tan sencillas. A medida que aumenta la cantidad de condiciones a comprobar, la estructura de árbol necesaria para tratar con ellas se vuelve inevitablemente más compleja. No obstante, y ésta es una de las grandes ventajas de utilizar este método para comprobar condiciones, los árboles más complejos no ocupan necesariamente más espacio en su programa.

Los manipuladores de personajes pueden ser tan complejos como pueda hacerlos, y el mejor enfoque consiste en desmenuzar el problema en módulos diferentes, tal como hemos hecho con el cuerpo principal del programa. La primera zona del comportamiento de los personajes que veremos es la forma en que se pueden manipular los objetos en el Dog and Bucket.

Rutinas necesarias y adecuadas

Precisamos, cuanto menos, rutinas para coger, soltar, comer, beber, entregar, recibir y arrojar objetos, y, por supuesto, necesitamos decirle al jugador (si estuviera presente) lo que está sucediendo.

Las rutinas "soltar, arrojar, entregar", por ejemplo, se podrían considerar como un grupo, porque todas ellas dependen de que el personaje comience con un objeto y termine sin él. Para determinar qué rutina es la más adecuada, obviamente necesitamos comprobar condiciones tales como "¿Hay alguien en la habitación?" y, si así fuera, "Ese alguien puede/quiere recibir el objeto?". Asimismo, necesitaríamos comprobar el humor del personaje, para ver si es probable o no que se arrojen objetos, etc. Otras complicaciones incluyen la necesidad de introducir en el proceso un elemento aleatorio, y la necesidad de otras rutinas relacionadas con el mismo (rebajar la energía de un personaje si, p. ej., uno de ellos ha sido golpeado con un vaso de cerveza). En el próximo capítulo dedicado a la programación de personajes interactivos diseñaremos un árbol que tendrá en cuenta todas estas posibilidades.

Complementos al BASIC

Spectrum:

Los complementos para las líneas 4000-4140 ya se han ofrecido en la página 1508. Los usuarios del Spectrum habrán de introducir los siguientes cambios adicionales:

```
10 h$="humano ":a$="animal ":m$="macho
   ":f$="hembra ":q$="Eres un ":DIM
   c(4),t(6,2)
40 PRINT q$;h$;:INPUT i$:c(1)=ABS(i$="s" OR
   i$="S")+1
50 PRINT q$;a$;:INPUT i$:c(2)=ABS(i$="s" OR
   i$="S")+1
60 PRINT q$;f$;:INPUT i$:c(3)=ABS(i$="s" OR
   i$="S")+1
70 PRINT q$;m$;:INPUT i$:c(4)=ABS(i$="s" OR
   i$="S")+1
```

Últimos retoques

Finalizamos nuestro proyecto con algunas sugerencias para perfeccionarlo

Los programas que presentamos se podrían mejorar de muchas formas. Consideraremos algunas de ellas antes de analizar los métodos que utilizan los programas de *go* en los grandes sistemas, algunos de los cuales probablemente se podrían implementar en microordenadores.

Los usuarios del Spectrum, Commodore 64 y Amstrad probablemente habrán notado que estos programas son traducciones de la versión original para el BBC Micro. Aunque en muchos casos se han modificado de forma significativa en razón de las diferencias en cuanto a instrucciones y estructura del programa, se han mantenido intencionadamente lo más similares posible. Su reescritura, utilizando nombres de variables más cortos y sacando mejor provecho de las facilidades de que disponen las distintas máquinas, podría mejorar su rendimiento de modo significativo.

Es poco probable que las versiones para el Spectrum y el Commodore 64 lleguen nunca a funcionar tan rápidamente como la versión para el BBC Micro, debido a que sus BASIC son más lentos. No obstante, quizá desee tratar de convertir algunas rutinas a lenguaje máquina, para acelerar la ejecución. La frecuencia con la que se ejecutan las rutinas se puede comprobar fácilmente colocando una variable de contador al comienzo de cada una, imprimiéndola luego al final del juego. Si hace esto, encontrará que una de las rutinas más críticas es PROCbuscar, que no sólo se utiliza con frecuencia durante el PROCevaluacion de grupos inicial, sino que también se llama al menos una vez para cada llamada a FNlegalidad. Teniendo el tablero (tablero%) establecido como una secuencia de bytes, la conversión a lenguaje máquina resultará bastante sencilla.

En el juego del *go* existen muchas otras tácticas básicas cuya inclusión ciertamente mejoraría el juego del programa. Entre ellas se incluyen configuraciones tales como escaleras, *joseki* (juego de apertura), etc. Una facilidad muy importante que se podría implementar es la de "vida y muerte". El programa ya incorpora una evaluación de "muerte incondicional". Ésta comprueba simplemente si a algún grupo determinado de fichas ya no le quedan licencias, suprimiéndolo en consecuencia del tablero. Dando un paso más hacia adelante, podría fácilmente juzgar si un grupo posee "vida incondicional". Para implementar esto deberíamos:

1. Llenar temporalmente tantas licencias del grupo como fuera legalmente posible con fichas del color contrario.

2. Utilizar luego PROCbuscar para contar las restantes licencias del grupo. Si este valor (clib%) es dos o más, el grupo no se puede capturar; a menos, por supuesto, que el defensor sea suficientemente torpe como para llenar sus propias licencias.

Hay que notar que cuando se llenan licencias deben probarse todas hasta el agotamiento. Si se prueban por orden, puede que no se llenen ojos falsos. Por ejemplo, el grupo quizá tenga una licencia interna que se pueda llenar sólo cuando se hayan llenado todas las licencias exteriores. Habiendo llenado una licencia interna, quizá sea posible llenar otros ojos falsos y verdaderos.

Habiendo implementado la vida incondicional en un grupo, puede pasar a la vida de un grupo en todo el tablero. Esto permite la posibilidad de grupos que compartan ojos. Encontrará un ejemplo de esto si retrocede hasta el primer capítulo de la serie. Para implementar esto, primero debe seguir el procedimiento anterior. De hallar una licencia interna que esté rodeada al menos por un lado por un grupo amistoso diferente, luego la rutina se debe llamar a sí misma recursivamente para comprobar la seguridad de este grupo adyacente, y así sucesivamente. El ojo interno del primer grupo sólo estará a salvo si los grupos adyacentes están a salvo de captura, lo que incide en la seguridad del grupo original.

Si bien el programa se podría mejorar significativamente de muchas maneras, es improbable que llegue a convertirse en un jugador de categoría internacional. Los programas que se ejecutan en grandes sistemas juegan apenas un poco mejor que un recién iniciado, aun después de más de veinticinco años de investigación. Ello se debe a numerosas razones.

Bien guardado

Una útil característica adicional sería la de poder guardar la partida en cinta o disco y retomarla tiempo después. Esto implicaría escribir una nueva rutina PROCguardar_partida, a llamar después de que un jugador digite DEJAR. El procedimiento para guardar tendría que guardar todas las variables de estado del juego y los valores de la tabla de bytes. Ésta se compone de:

tablero% to tablero% + 255	Bytes principales del tablero
estimaciones% to estimaciones% + 255	PROCsuprimir puede haber cambiado los valores de la tabla de estimaciones
movimiento%	No sólo da el número de movimiento, sino que indica el siguiente jugador: impar o par
captura%(2)	Cantidad de fichas capturadas por negras y blancas
ko%	Indica la posición de un punto *ko*, o cero

Asimismo, y de modo opcional, podría guardar:

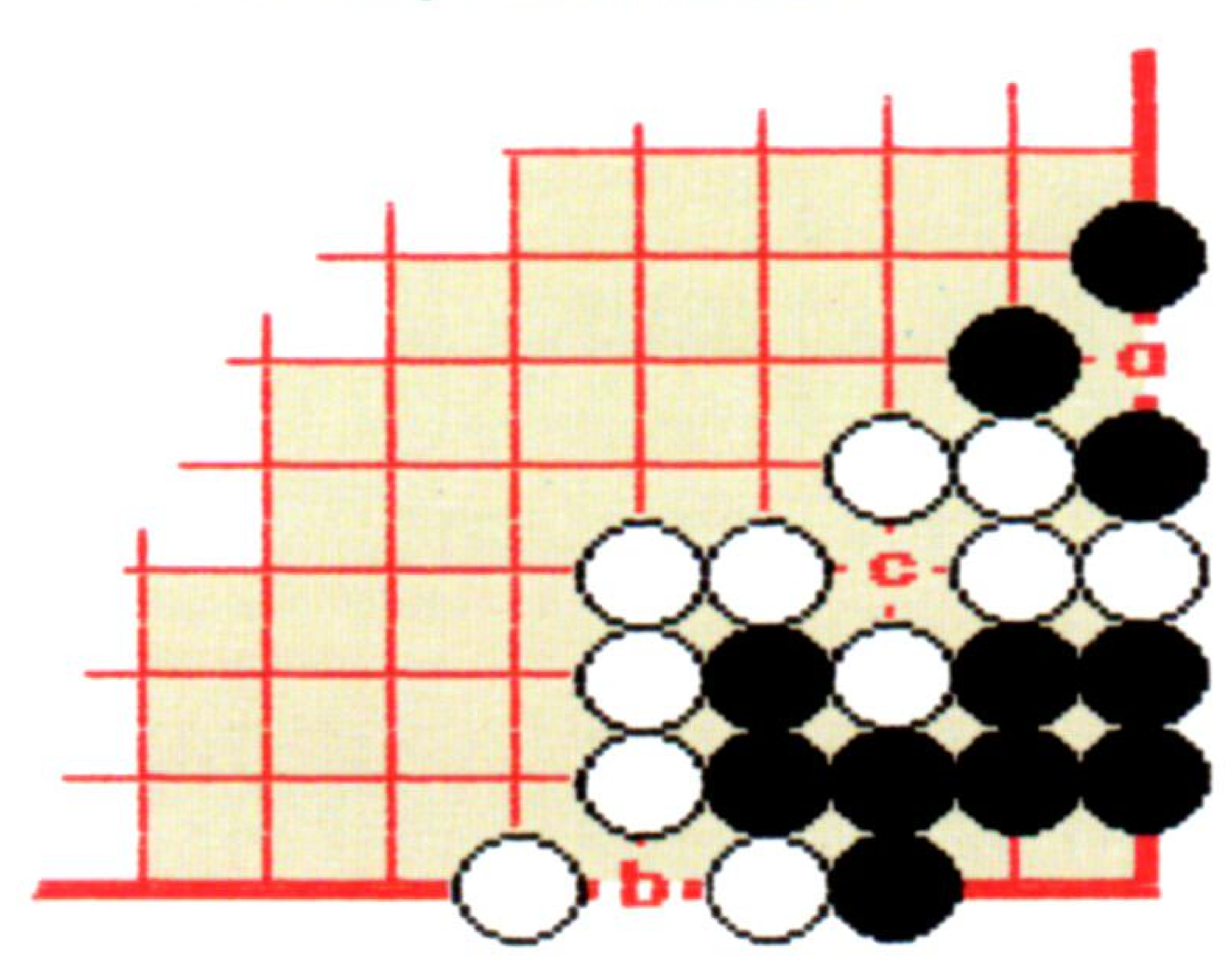
atari1$ y atari2$	Indican una advertencia habitual de "atari" para las negras o las blancas
T$	Método de la última elección de movimiento utilizado por las negras

El propio tablero es fuente de numerosos problemas en virtud de su tamaño. Por lo general se dice que un tablero de 19 por 19 tiene un orden de magnitud de 3^{361}, que es aproximadamente 10^{172}. Esta cifra se calcula como el límite superior de la cantidad de configuraciones de puntos negros o blancos vacantes del tablero. Una evaluación aproximativa de la cantidad media de movimientos potenciales por parte del jugador es 250. Si se aplicara al *go* la técnica tradicional para juegos de la búsqueda arborescente, una búsqueda exhaustiva con sólo tres movimientos de anticipación supondría la generación y evaluación de alrededor de 8 000 000 de posiciones del tablero. Pero informes sobre el juego grabados en cinta de video han revelado que los jugadores aficionados avanzados pueden anticipar la sorprendente cantidad de 30 movimientos, y la literatura sobre *go* con frecuencia contiene secuencias anticipadas aún más profundas.

La solución al problema de los árboles grandes consiste en utilizar algún tipo de función de "enfoque". Ésta efectuaría inicialmente una evaluación aproximativa de todo el tablero, al objeto de decidir qué zonas del tablero se hallan en situación crítica. Habiendo elegido una o dos zonas pequeñas, se pueden generar árboles de profundidad total sólo para estos patrones de fichas. Asimismo, los árboles se pueden *podar* utilizando el algoritmo alfabeta y proporcionando rutinas que puedan captar los puntos críticos, las posiciones de los ojos, etc.

Otro de los problemas del *go* es la imprecisión de sus reglas. Por ejemplo, el final del juego se determina cuando ninguno de los jugadores puede obtener ninguna otra ventaja, pero los programas de *go* tienen dificultades para determinar esta condición más bien vaga. Otro problema está relacionado con las condiciones *ko* especiales. Por ejemplo, el diagrama muestra una posición de *triple ko*. Las negras juegan en "b", capturando la ficha blanca de la derecha. Entonces, las blancas capturan la ficha negra situada justo abajo de "a", tras lo cual las negras hacen una captura en "c". Ahora las blancas pueden volver a jugar en la posición de la primera captura, eliminando la ficha negra situada en "a". Las negras recapturan la ficha blanca de "a", y así sucesivamente. En el transcurso de una partida normal, si ninguno de los jugadores está deseoso de abandonar la lucha local jugando en algún otro sitio, entonces el juego se cancela. Aunque no es habitual, se pueden producir situaciones aún más complejas. Para reconocerlas, un programa de *go* habrá de almacenar todas las posiciones del juego anteriores de modo que puedan cotejarse con la posición actual.

En la actualidad los programas de *go* no juegan bien. No obstante, hace muy pocos años eran pocas las personas que reconocían que los ordenadores llegarían algún día a jugar una buena partida de ajedrez. Ahora la cuestión es si un programa de ajedrez por ordenador llegará alguna vez a alcanzar el estatus de campeón del mundo.



Marcador al final del juego



Al final del juego se debe evaluar el tablero y otorgar puntos a cada parte según la cantidad de territorio capturado.

El grupo blanco de la esquina inferior derecha del tablero ha rodeado 17 puntos de territorio. No se incluyen las dos fichas blancas dentro de esta zona: sólo se cuentan los puntos vacantes.

El grupo negro de la esquina superior derecha en realidad son dos grupos, ya que entre las dos mitades sólo hay una «conexión» en diagonal. Sin embargo, todas las salidas están cortadas por fichas negras y, por lo tanto, las negras pueden sumar 17 a su marcador.

Las blancas rodean la esquina inferior izquierda, pero aquí hay una ficha negra suelta. Las reglas del *go* afirman que en realidad no es necesario capturar esta ficha durante el juego. Es en este punto cuando se elimina la ficha, dejando 10 puntos de territorio para las blancas. Al marcador de éstas se le suma un punto para dar cuenta de la ficha negra eliminada.

El bando de las negras parece haber capturado la esquina superior izquierda, pero él mismo se halla rodeado y cortado por un grupo de blancas más numeroso. En consecuencia, se puede eliminar del tablero al grupo de negras y las blancas consiguen 18 puntos territoriales y otros 5 más por las fichas negras capturadas. Todos los otros puntos vacantes son neutrales, porque no forman zonas de territorio rodeadas exclusivamente por ninguno de los colores

Caroline Clayton

Datos básicos (IX)

Por cortesía de la Commodore Business Machines, reproducimos otro fragmento del mapa de memoria del Commodore 64

ETIQUETA	DIRECCIÓN HEXA	POSICIÓN DECIMAL	DESCRIPCIÓN
SAREG	030C	780	Almacenamiento para 6502. Registro A
SXREG	030D	781	Almacenamiento para 6502. Registro X
SYREG	030E	782	Almacenamiento para 6502. Registro Y
SPREG	030F	783	Almacenamiento para 6502. Registro SP
USRPOK	0310	784	Instrucción salto función USR (4C)
USRADD	0311-0312	785–786	Byte *lo*/byte *hi* de la dirección USR
	0313	787	No utilizado
CINV	0314-0315	788-789	Vector: Interrupción hardware IRQ
CBINV	0316-0317	790-791	Vector: Interrupción instrucción BRK
NMINV	0318-0319	792-793	Vector: Interrupción no enmascarable
IOPEN	031A-031B	794-795	Vector rutina núcleo OPEN
ICLOSE	031C-031D	796-797	Vector rutina núcleo CLOSE
ICHKIN	031E-031F	798-799	Vector rutina núcleo CHKIN
ICKOUT	0320-0321	800-801	Vector rutina núcleo CHKOUT
ICLRCH	0322-0323	802-803	Vector rutina núcleo CLRCHN
IBASIN	0324-0325	804-805	Vector rutina núcleo CHRIN
IBSOUT	0326-0327	806-807	Vector rutina núcleo CHROUT



El gran competidor

Inmerso en el entorno GEM y basado en el procesador Motorola 68000, el flamante Atari 520ST parece anunciar una nueva era para los microordenadores

Chris Stevens

Es probable que 1985 se constituya en una especie de línea divisoria para la industria del microordenador personal, el año en el que el crecimiento casi exponencial de años anteriores comenzó a disminuir y la industria empezó a madurar. Esto se puede deducir a partir de los problemas dados a conocer públicamente por varios fabricantes de microordenadores, que, a su vez, han fijado un definido margen de precaución en el área de marketing: la mayoría de los productos nuevos han sido versiones remozadas de máquinas de éxito, como el BBC B+, el Commodore 128 y el Atari 130XE.

Sin embargo, 1985 también podrá recordarse como el año en que Atari resurgió de las cenizas para volver a establecerse como uno de los principales fabricantes de micros personales. Esto se debe, en parte, a la influencia del nuevo propietario de la empresa, Jack Tramiel, quien no sólo aportó una aguda perspicacia comercial, sino que también generó, como uno de los "personajes" de la industria, un gran interés en los medios de comunicación.

Por sí solo esto no habría sido suficiente para salvar a Atari, dado que su principal problema era su línea de productos, que, esencialmente, se consideraba pasada de moda. El lanzamiento del Atari 520ST cambia radicalmente esta situación. El ordenador está diseñado alrededor del procesador de 16 bits Motorola 68000, el mismo utilizado en el Apple Macintosh. Este chip por lo general se considera el más avanzado de los disponibles actualmente en grandes cantidades, con arquitectura interna a gran escala de 32 bits, 17 registros de 32 bits, un bus de datos de 16 bits y un bus de direcciones de 24 bits. Obviamente, en base a estas especificaciones se puede diseñar un ordenador muy potente, que es precisamente lo que ha hecho Atari.

Producido en el mismo estilo y aspecto que el 130XE, el 520ST tiene un acabado de resistente plástico gris. El teclado está dividido en cuatro secciones, encima de las cuales hay 10 teclas de función programable, empotradas en la carcasa como las del 130XE. Debajo de éstas hay un teclado estándar, con la adición de una tecla Alternate. Cuando no está conectado el ratón, ésta se utiliza para controlar el cursor de pantalla.

A la derecha hay un racimo de cursor, junto con otras teclas para el editor de pantalla, como Clear e Insert. Justo arriba del racimo del cursor están las teclas Help y Undo y, a la derecha del teclado, hay un relleno numérico con teclas para las funciones aritméticas.

Alejándose de su práctica anterior, Atari ha adoptado para el ordenador varias interfaces "estándares"; las máquinas anteriores de Atari tenían sus propias puertas en serie para todos los periféricos. El Atari 520ST tiene una interface en paralelo Centronics para conexión a impresoras y una puerta en serie RS232 tipo D de 25 vías para la instalación de un modem u otro dispositivo en serie.

Lo más interesante es que el ordenador se ha equipado con una interface MIDI. Ésta está insta-

A la reconquista del mercado
El 520ST representa el intento de Atari por recuperar el lugar que ocupó en el mercado de microordenadores a finales de los años setenta. Basado en el procesador Motorola 68000, el ordenador posee numerosas configuraciones avanzadas, como 512 Kbytes de RAM, puertas MIDI y el sistema operativo GEM. Hasta ahora, el GEM sólo había estado disponible en máquinas mucho más caras

lada como un par de conectores DIN (para MIDI IN y MIDI OUT), lo que significa que el ordenador puede controlar directamente la operación de sintetizadores y otros instrumentos musicales. No obstante, las puertas MIDI tienen otros usos. Puesto que la MIDI es un dispositivo en serie rápido (31,25 Kbaudios), también se puede emplear como método alternativo para la transmisión de datos entre ordenadores. Atari pretende sacar partido de esta facilidad para producir un sistema en red de área local (LAN) utilizando las puertas MIDI.

Otra facilidad interesante que se le ha añadido al nuevo ordenador es una interface para "disco rígido". En la actualidad Atari desea estar entre los primeros en lanzar un reproductor "CD-ROM". Éste es un desarrollo del disco compacto que permitirá almacenar hasta 800 Mbytes de información en un único disco. Por supuesto, ahora mismo los discos compactos son dispositivos de lectura solamente, pero su potencial es extraordinario.

En el precio de la nueva máquina Atari se incluye una unidad de disco, que se enchufa en la única interface para disco flexible. Si se requiere otro disco, se puede enchufar en la primera unidad de disco. No se proporciona puerta para cassette. Atari ha optado por los discos Sony de 3 ½ pulgadas, que se están haciendo sumamente populares para una amplia gama de micros. Las unidades de disco se están produciendo en versiones de una sola cara o de doble cara que, al formatearlas, proporcionan 320 K por cada cara.

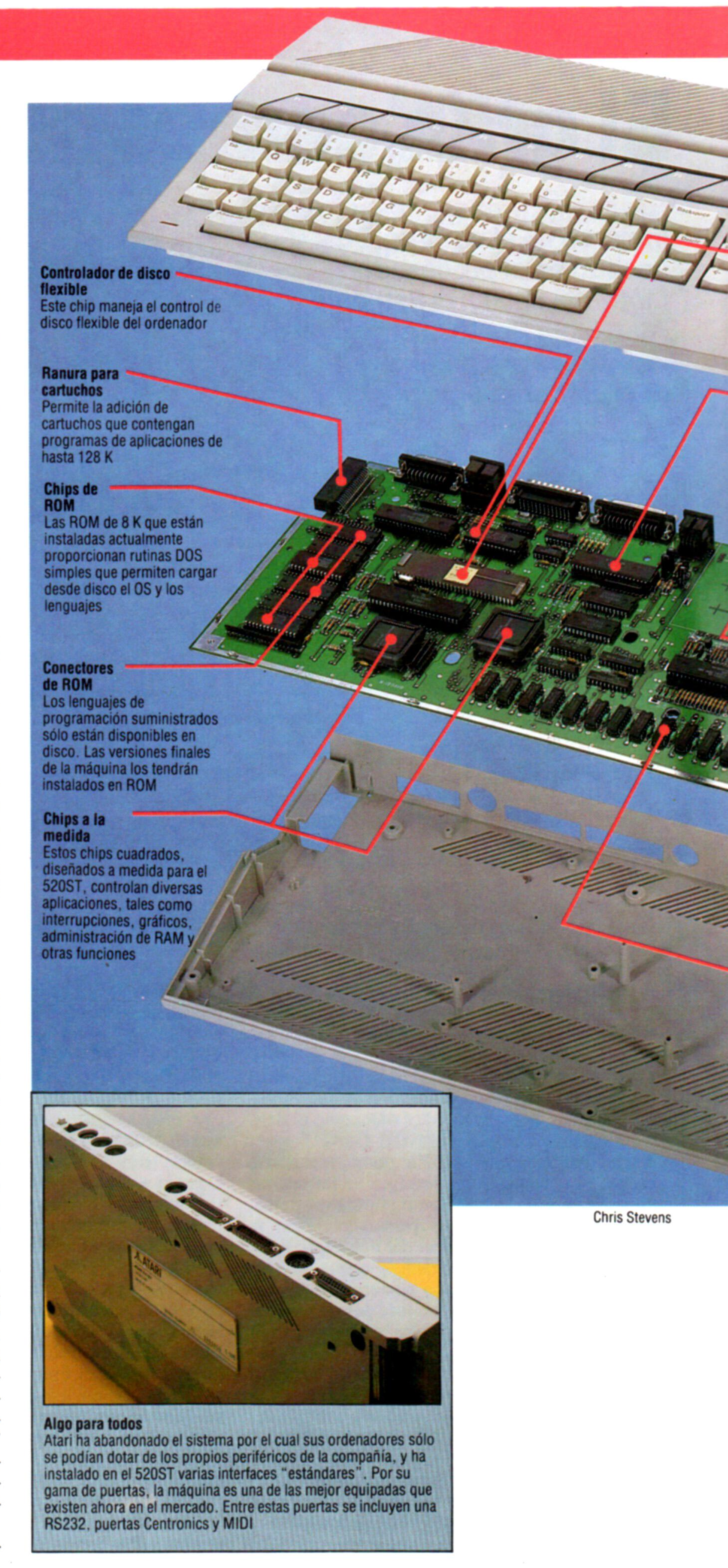
Completando la lista de interfaces que vienen en el ordenador hay un conector de potencia, un monitor RGB y puertas RF (en futuras versiones de la máquina), una puerta para cartuchos capaz de utilizar ROMs de 128 K, y un par de puertas para palanca de mando sobre el lado derecho, aunque éstas no están pensadas específicamente para tal fin, sino más bien para controladores de ratón.

Entorno GEM

Ésta es la última característica que explica el entusiasmo que ha despertado este ordenador. El 520ST es el primer micro de precio reducido que se introduce con el entorno GEM como extremo frontal estándar para el sistema operativo. Desarrollado por Digital Research, el GEM (siglas de *Graphics Environment Manager*: administrador del entorno gráfico) proporciona un sistema WIMP (*Windows, Icons, Mouse Program*), como el que ha obtenido tanto éxito en el Apple Macintosh. De hecho, el sistema GEM, tal como está implementado en el nuevo micro Atari, guarda una sorprendente semejanza con el del Macintosh.

El control del sistema se proporciona desplazando un cursor de flecha (que cuando la máquina está "atareada" se convierte en el icono de una "abeja") por la pantalla a varios iconos. Pulsando el botón del ratón se selecciona el icono determinado. En la parte superior de la pantalla también hay disponibles unos menús que se pueden "bajar" como ventanas. Por ejemplo, una unidad de disco se selecciona eligiendo el icono del "mueble archivador" situado en la esquina superior izquierda de la pantalla inicial. Luego se pueden realizar diversas manipulaciones de archivo tirando del menú "archivos"; "opciones" visualiza el directorio.

El GEM visualiza dos tipos de archivo. Los ico-



Chris Stevens

Algo para todos
Atari ha abandonado el sistema por el cual sus ordenadores sólo se podían dotar de los propios periféricos de la compañía, y ha instalado en el 520ST varias interfaces "estándares". Por su gama de puertas, la máquina es una de las mejor equipadas que existen ahora en el mercado. Entre estas puertas se incluyen una RS232, puertas Centronics y MIDI





nos "carpeta" denotan "directorios", mientras que los iconos cuadrados representan archivos de programas. El programa se carga (LOAD) y ejecuta (RUN) con sólo mover el cursor hasta el archivo. Si quiere borrar un archivo, sólo tiene que desplazar el icono en cuestión hasta el icono de la "papelera". La facilidad con que hasta un novato puede aprender a emplear el GEM es particularmente evidente para cualquiera que haya utilizado alguno de los sistemas operativos más convencionales, como el CP/M y el MS-DOS.

El 520ST dispone de tres modalidades para gráficos: alta, media y baja resolución. La modalidad más alta posee una resolución de 640 por 400 pixels, pero sólo aparece en monocromático, lo que le confiere a la máquina un aspecto aún más similar al del Mac. Las modalidades de menor resolución demuestran las capacidades de color de la máquina, pero muchos usuarios preferirán la modalidad monocromática de alta resolución. Atari está ofreciendo para el 520ST la opción de una pantalla monocromática o bien un modelo en color.

El entorno de "escritorio" del GEM (así llamado porque uno desplaza los iconos de forma muy similar a como movería los papeles por la superficie de un escritorio), opera bajo TOS (*Tramiel operating systems*: sistema operativo Tramiel). Ésta es una versión construida a la medida del sistema operativo CP/M 68 de Digital Research, en sí mismo un desarrollo del popular OS de ocho bits de Digital Research, que se ha vuelto a diseñar para máquinas basadas en el 68000. Sin embargo, a diferencia del CP/M 68, el sistema operativo permite organizar archivos en "directorios" como el MS-DOS, el sistema operativo de 16 bits que es el estándar *de facto*.

El 520ST tiene empaquetados algunos programas de aplicaciones. Primero está *GEM Write*, un paquete de tratamiento de textos, y *GEM Paint*, que, como su nombre sugiere, es un paquete para diseñar gráficos. Similar a los equivalentes *MacPaint* y *MacDraw* del Macintosh, el paquete ofrece varios tamaños de "pincel" y diseños de "relleno", así como una facilidad de *zoom* que permite que el usuario enfoque una pequeña zona de la pantalla con el fin de producir un trabajo especialmente minucioso. Si bien el *GEM Paint* carece de algunas de las facilidades del Macintosh, es, así y todo, un programa muy sofisticado.

Lenguajes basados en ROM

En el ordenador se incluyen también los lenguajes de programación ST-BASIC y ST-LOGO, que en las primeras máquinas se proporcionaban en disco, pero en ulteriores versiones aparecerían en ROM en la tarjeta. Como el TOS, los lenguajes son desarrollos de productos anteriores de Digital Research; el BASIC es la versión propia de DR del BASIC Microsoft, conocido como Personal BASIC, mientras que el LOGO es una versión del popular Dr LOGO, ahora incluido en algunas máquinas. Los lenguajes han sido adaptados para sacar partido de las capacidades del entorno GEM para ventanas.

El Atari 520ST es indudablemente una máquina importante, no sólo para la propia Atari Corporation, sino también para el mercado de ordenadores "económicos". El sistema completo incluye una unidad de disco, una pantalla monocromática y un ratón, así como el software empaquetado. Por su precio está al alcance del usuario personal "serio" y el de pequeña gestión, aunque tal vez sea demasiado caro para generar ventas muy grandes, de las que depende el esencial apoyo de software.

Junto con los paquetes GEM y los 512 K de memoria, el usuario obtiene una configuración similar al *Fat Mac* de Apple por la mitad de precio. En el caso de que saliera una versión más pequeña del ordenador, con, por ejemplo, 256 K de memoria, las ventas resultantes serían muchísimo mayores. Sin embargo, una cosa es cierta: cuando el 520ST esté disponible en cantidades suficientes como para hacer un impacto en las zonas comerciales, el mercado de ordenadores económicos habrá entrado en una nueva era.

ATARI 520ST

DIMENSIONES
470 × 240 × 60 mm

CPU
Motorola 68000 trabajando a 8 MHz

MEMORIA
512 Kbytes de RAM

PANTALLA
Tres modalidades de resolución: la más alta es la modalidad monocromática, con 640 × 400 pixels

INTERFACES
MIDI (dos conectores DIN), interface en paralelo Centronics, puerta en serie RS232, interface para disco flexible, conector de potencia, monitor RGB, RF, puertas para cartuchos y palanca de mando. También hay una interface para "disco rígido" para futuras reproductoras de CD-ROM

UNIDAD DE DISCO
Una unidad de disco Sony de 3 ½ pulgadas y doble cara

SISTEMA OPERATIVO
TOS y GEM como estándar

LENGUAJES DISPONIBLES
ST-BASIC y ST-LOGO

TECLADO
84 teclas más 10 teclas de función

DOCUMENTACION
El manual del usuario está muy bien elaborado, con un gran número de diagramas e instantáneas de pantallas acompañando un texto conciso

VENTAJAS
El ordenador ofrece tecnología de 16 bits, mucha memoria para el usuario y un sistema operativo amable con el usuario que previamente sólo había estado disponible en máquinas mucho más caras

DESVENTAJAS
A pesar de sus notables especificaciones, el sistema aún ha de probarse a sí mismo, y quizá el ordenador sea aún demasiado caro como para generar grandes ventas, de las que depende el vital soporte de software

Punto a punto

Comenzamos la construcción del tester ocupándonos del montaje de los componentes pasivos

Lista de componentes

Cant.	Ref.	Artículo
1	C1	condensador de polipropileno de 0,47 μF (1)
2	C2-3	condensador de policarbonato de 1,0 μF
1	C4	condensador de policarbonato de 0,1 μF
1	C5	condensador de poliestireno de 470 pF
5	C6	condensador de disco cerámico de 22000 pF
1	D1	diodo de ref. de 1,22 V de intervalo de banda 9491 (2)
1	D2	diodo 1N4148
6	TR1-5,TR7	transistor NPN 2N2219
1	TR6	transistor PNP 2N2905
1	VR1	potenciómetro cermet multivueltas de 10 k-ohmios
1	VR2	potenciómetro preajustado cermet de 10 k-ohmios
1	IC1	chip convertidor A/D 7135
1	IC2	temporizador de baja potencia ICM7555
1	IC3	activador de 7-segmentos 7447A
1	—	zócalo DIL de 28 patillas
2	—	zócalo DIL de 16 patillas
1	R1	resistencia de 8,2 k-ohmios (3)
7	R2-8,R14,R17	resistencia de 180 ohmios
1	R9	resistencia de 47 k-ohmios
3	R10-12	resistencia de 100 k-ohmios
1	R13,R15,R16	resistencia de 4,7 k-ohmios

Notas:
1) El condensador C1 puede sustituirse por uno de policarbonato (con alguna pérdida de prestaciones)
2) Este diodo puede sustituirse por un diodo zener de 2,7 V (con alguna pérdida de prestaciones)
3) Se deben utilizar resistencias de 5 % de tolerancia, de carbón o de película metálica

Para la construcción del módulo básico recomendamos utilizar una placa matriz de topos de 0,1 pulgadas en lugar de las placas de circuito impreso de tiras. Esto se debe a que éstas comprometen el trazado del cableado una vez y para siempre, mientras que el cableado punto por punto de una placa de topos permite la introducción de ligeras modificaciones en el posicionamiento de las conexiones para optimizar el rendimiento.

A diferencia de los circuitos digitales comunes, un convertidor A/D tiene entradas analógicas muy sensibles (alta impedancia de entrada). Esto significa que las señales de interferencia y no deseadas pueden introducirse en las entradas analógicas a menos que se tomen medidas especiales. En nuestro diseño todo el sistema de circuitos analógico se mantiene en uno de los lados del chip convertidor A/D (patillas de 1 al 10), y todo el sistema de circuitos digitales en el otro. Aun así, durante la construcción del prototipo encontramos que unas ligeras modificaciones en la posición del cableado podrían mejorar de modo significativo el rendimiento del circuito. Idealmente, tal circuito utilizaría una placa de circuito impreso comprobada, pero su construcción está fuera del alcance del montador medio, de modo que hemos optado por un cableado punto a punto.

El esquema del cableado muestra una vista desde arriba de la placa del circuito; los conductores de interconexión estarán en la cara inferior de la placa. El trazado del cableado está muy estilizado para que resulte más claro, mostrando sólo algunos conductores conectados directamente de un punto a otro.

Este trazado angular responde sólo a razones de claridad y no es necesario respetarlo exactamente. A pesar de que ofrecería un mejor aspecto, es mejor no montar los conductores muy tirantes entre dos puntos, para poder desplazarlos más adelante para optimizar el rendimiento global.

La placa prototipo se cableó empleando terminales para hilo arrollado (*wire-wrapping*) y una herramienta de arrollado especial para conectarlos. Esta técnica es muy conveniente, porque si usted conecta un cable de modo erróneo, puede desarrollarlo y volverlo a conectar con toda facilidad. La alternativa a este método es utilizar hilo de conexión delgado corriente y un soldador.

La mayoría de los chips semiconductores tienen patillas con 0,1 pulgadas (2,45 mm) entre centros, de modo que la placa de circuito más adecuada es una placa matriz de 0,1 pulgadas con cada orificio rodeado por un topo de cobre para realizar la soldadura. Existen versiones sin cobre de este tipo de placa, pero indudablemente es más fácil trabajar con el tipo que tiene topos de cobre.

Para que haya amplio espacio para los componentes, la placa matriz ha de tener alrededor de 50 filas por 45 columnas de agujeros. Comience por insertar los tres zócalos de IC: el zócalo DIL de 28 patillas para el convertidor A/D 7135 (ICI), un zócalo de ocho patillas para el chip temporizador 7555 (IC2) y un zócalo de 16 patillas para el activador de siete segmentos 7447A (IC3). Si se utiliza una placa matriz con cobre, suelde dos patillas en esquinas opuestas de cada zócalo DIP por la parte posterior de la placa. Si está utilizando la placa matriz de tipo sin cobre, una pequeña gota de soldadura en las patillas de las esquinas mantendrá a los zócalos en su sitio.

Antes de montar ningún otro componente, suelde o arrolle las líneas de +5 V (Vcc) a los tres IC. Por razones de claridad, no se muestra la línea Vcc completa, pero finalmente todos los puntos de +5 V (como la patilla 16 de IC3) se conectarán juntas y serán alimentadas por la fuente de alimentación. Sólo un chip, el 7135, necesita una fuente de −5 V (Vee). Ésta va desde la patilla 1 de IC1 hasta un punto adecuado en el borde de la placa principal, como se indica.

Puntos de masa

Se necesitan dos puntos de masa separados: uno para la masa analógica (situado junto a la patilla 4 de IC1) y uno para la masa digital. Como ya hemos dicho anteriormente, es importante conectar todas las masas analógicas a un punto único de masa analógica, y todas las masas digitales a otro punto único de masa digital. Conecte la masa analógica de IC1 (patilla 3) a la masa analógica y las masas de IC2 e IC3 (patillas 1 y 8, respectivamente) a la masa digital. La masa digital de IC1 (patilla 24) también se conecta al punto de masa digital.



Disposición de los componentes

A LAS SEÑALES DE HABILITACIÓN DE LOS 7 SEGMENTOS

A LOS VISUALIZADORES LED DE 7 SEGMENTOS

A PATILLAS 9-15 DE IC3

TR6 TR5 TR4 TR3 TR2 TR1

A R7
A R8
A R2
A R3
A R4
A R5
A R6

MASA DIGITAL

MASA ANALÓGICA

VR2

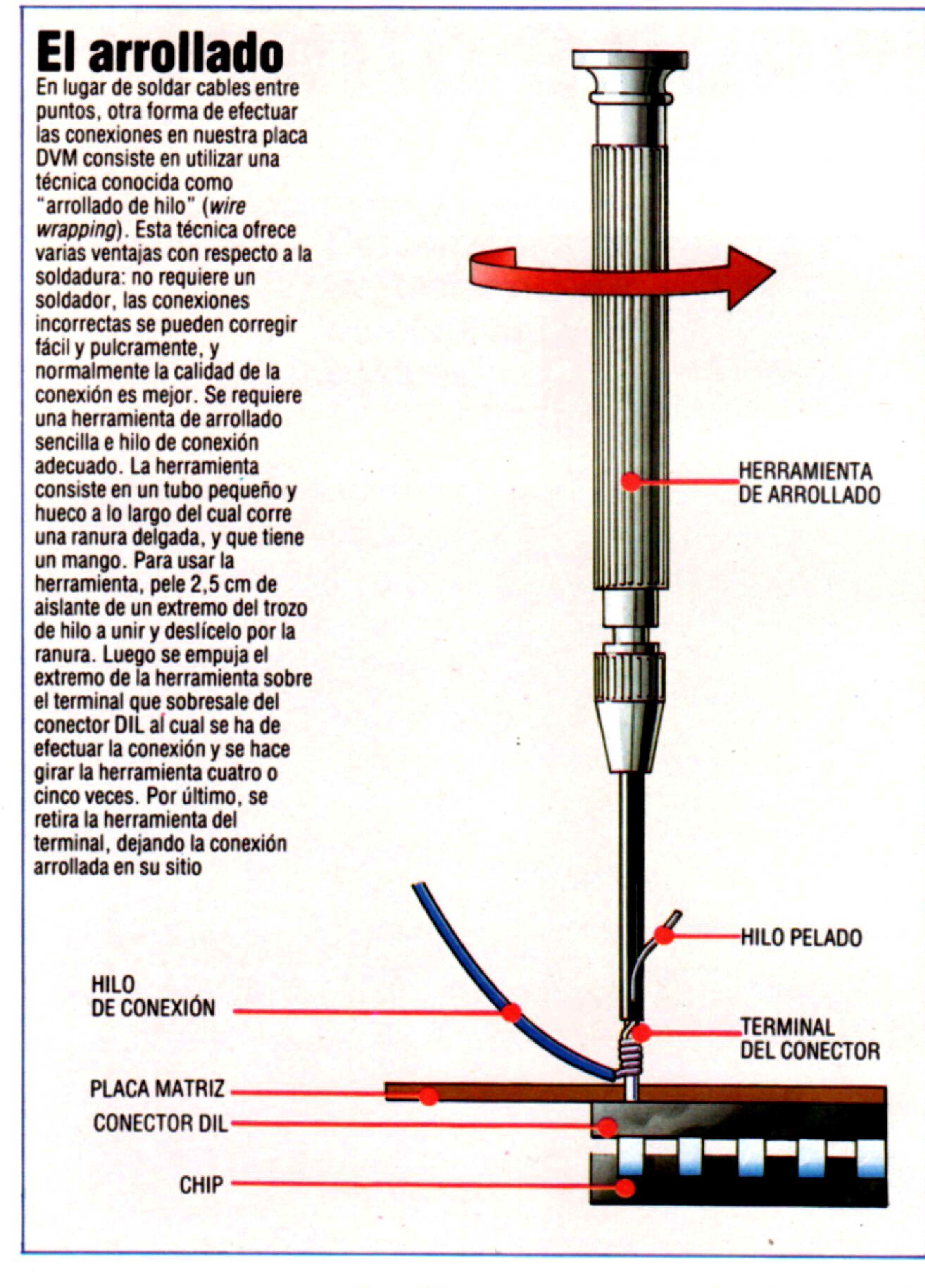
Kevin Jones

Montaje de componentes

En el esquema del montaje vemos los componentes montados en una placa matriz de 0,1 pulgadas. El cableado punto a punto se muestra en estilo angular para facilitar la comprensión. En la práctica, el cableado entre puntos de la placa debe quedar suficientemente suelto como para permitir el movimiento de los cables, ya que la posición del cableado puede incidir en el rendimiento del tester. Observe que no se indican explícitamente las conexiones entre puntos analógicos y digitales a masa, de D5 a la masa digital y todas las líneas de alimentación de +5 V. La conexión entre las masas analógica y digital se debe efectuar con hilo de cobre bastante grueso. El método más simple de proporcionar las alimentaciones de +5 V consiste en conectar todas estas conexiones a un único trozo de cable que corra alrededor del borde de la placa. Como se menciona en el texto, si las tensiones a medir son con respecto a masa, se debe conectar la patilla 9 del chip 7135 al punto de masa analógica. Suelde todos los componentes pasivos en su sitio, tal como se indica, pero todavía no monte los chips en sus conectores, porque éstos pueden dañarse fácilmente. En el próximo capítulo nos ocuparemos de esta parte de la construcción. Asimismo, se deben conectar condensadores de disco de 22 000 pF entre las fuentes de alimentación del chip y masa. En el IC1 conecte un condensador entre la patilla 1 y el punto de masa analógica y otro entre la patilla 11 y la masa analógica. En el IC2 conecte un condensador entre la patilla 8 y la masa digital, y en el IC3 entre la patilla 16 y la masa digital. También se debe conectar un condensador electrolítico de 47 μF entre la patilla 1 de IC1 y la masa analógica. Observe que en este tipo de condensador la orientación tiene importancia. Asegúrese de conectar el terminal negativo del condensador a la patilla 1 del chip y el lado positivo a masa

El arrollado

En lugar de soldar cables entre puntos, otra forma de efectuar las conexiones en nuestra placa DVM consiste en utilizar una técnica conocida como "arrollado de hilo" (*wire wrapping*). Esta técnica ofrece varias ventajas con respecto a la soldadura: no requiere un soldador, las conexiones incorrectas se pueden corregir fácil y pulcramente, y normalmente la calidad de la conexión es mejor. Se requiere una herramienta de arrollado sencilla e hilo de conexión adecuado. La herramienta consiste en un tubo pequeño y hueco a lo largo del cual corre una ranura delgada, y que tiene un mango. Para usar la herramienta, pele 2,5 cm de aislante de un extremo del trozo de hilo a unir y deslícelo por la ranura. Luego se empuja el extremo de la herramienta sobre el terminal que sobresale del conector DIL al cual se ha de efectuar la conexión y se hace girar la herramienta cuatro o cinco veces. Por último, se retira la herramienta del terminal, dejando la conexión arrollada en su sitio



Por último, conecte entre sí los dos puntos de masa utilizando hilo de conexión bastante grueso. (Por razones de claridad, esta conexión no se ha ilustrado, ¡pero no olvide realizarla!)

Antes de instalar ningún otro componente, emplee un óhmetro o algún otro comprobador de continuidad sencillo para verificar la continuidad entre el punto de entrada de la fuente de alimentación y las patillas Vcc de los tres IC. De modo similar, compruebe la continuidad entre todos los puntos de masa.

Una vez montados los zócalos de los IC, se pueden insertar los otros componentes en las posiciones que se indican. Los tamaños exactos que se muestran son para unos determinados modelos de componentes especificados en la *Lista de componentes*. La mayor parte de ellos se pueden reemplazar por otros equivalentes, pero las dimensiones pueden ser diferentes y entonces habrá que modificar ligeramente la disposición. El componente más especial de todos es C1: el condensador integrador de 0,47 μF.

Para un mejor rendimiento, éste ha de ser de polipropileno. En el caso de que tenga dificultades para encontrarlo en el mercado, sustitúyalo por un condensador de policarbonato (que funcionará, aunque no tan bien), o bien encuentre un distribuidor de componentes electrónicos que lo pida a fábrica.

Otro componente que puede resultar problemático son los LED de siete segmentos. Los más corrientes son los dobles (ya sea de dos "8" o bien de un "+/−1" y un "8"), u "8" individuales, pero no "+/−1" individuales. Esto significa que, utilizando los componentes más corrientes, es imposible construir un visualizador de 4 ½ dígitos capaz de mostrar un auténtico signo más o menos sin tener un dígito sobrante.

En el mercado existen LED adecuados (empleados en el prototipo), pero por si hubiese dificultades para obtener estos componentes (y por su elevado precio), hemos modificado ligeramente el circuito original para hacerlo funcionar con LED de 0,5 pulgadas y dígito simple. Por tanto, se utiliza un LED de siete segmentos corriente para visualizar un vacío o un 1, y también para visualizar el signo menos utilizando el segmento horizontal central (segmento C) cuando la señal que se está midiendo es negativa. Las señales positivas se visualizarán sin ningún signo delante de ellas.

VR1 debe ser un potenciómetro de ajuste multivueltas de buena calidad; VR2 puede ser cualquier potenciómetro de corrección preajustado. Lo ideal es que D1 sea un diodo de referencia de intervalo de banda como uno del tipo 9491. Para evitar la incomodidad que entraña conseguir este componente, se puede sustituir por un diodo zener de 2,7 V. Nuevamente, este componente funcionará, pero con un menor rendimiento global.

Tipos de transistores

Los transistores (con la excepción del TR7) se utilizan para activar los visualizadores LED. Observe que los transistores del 1 al 5 son tipos NPN y que TR6 es de tipo PNP. Esto se debe a que TR6 ha de encender en respuesta a una señal LO lógica, mientras que los otros han de activar en respuesta a una señal HI lógica. TR7 se emplea en el circuito de borrado a cero. D2 puede ser un pequeño diodo de silicio cualquiera, si bien el componente especificado se puede obtener fácilmente.

Conecte los puntos de entrada a las patillas IC1 9 y 10, utilizando un trozo de cable trenzado o algún cable coaxial delgado (malla a la patilla 9, y central a la 10). En nuestro prototipo, se consiguieron resultados mucho mejores conectando la patilla 9 de IC1 a la masa analógica (que no se indica en el diagrama). Se aconseja realizar esta conexión si no se requiere una entrada de punto flotante, dado que las tensiones se miden con relación a masa y no como la diferencia entre tensiones positivas y negativas.

Antes de insertar los IC, compruebe concienzudamente su cableado comparándolo con el esquema presentado. Observe, por ejemplo, que TR6 no tiene aleta y que su colector y su emisor están conectados en sentido contrario a los de TR1 a TR5.

Próximamente nos ocuparemos del cableado del panel visualizador y la manipulación de los IC.

Una manipulación torpe destruirá rápidamente los IC, de modo que le aconsejamos que siga atentamente las instrucciones, a menos que esté usted totalmente familiarizado con el manejo de componentes CMOS.



Característica exclusiva

Llegados a este punto en nuestro curso, centraremos nuestra atención en la manera de ampliar el FORTH para tratar estructuras de matriz

Ya hemos visto algunas de las ideas que es necesario combinar para que uno defina sus propias matrices cuando utiliza CREATE en el programa *La criba de Eratóstenes*; equivalía a:

```
CREATE BITS 8192 ALLOT
```

En ese programa, CREATE incluye un encabezador en el diccionario empleando el nombre BITS desde el teclado, y también incluye un campo de código que hace que BITS, cuando usted la utiliza, deje en la pila la dirección de su campo de parámetros, pfa. No obstante, CREATE en sí misma no hace nada para establecer el campo de parámetros, y es aquí donde se introduce ALLOT. ALLOT toma un número de la pila e incluye en el diccionario espacio para el número de bytes correspondiente. Dado que en este caso se incluyen inmediatamente después del campo de código incluido por CREATE, constituyen el campo de parámetros para BITS.

Potencias de tres

He aquí un sencillo ejemplo que define una palabra, 3**, para calcular potencias de tres. Para que sea rápida, emplea una matriz que contiene todas las respuestas posibles, y se prepara utilizando CREATE con la palabra del FORTH, (una coma):

```
VARIABLE INDICEMAX
:POTENCIAS,(--)
      (incluye todas las potencias de tres que pueden
      caber en dos bytes, y coloca en INDICEMAX el
      índice de la potencia mayor.)
   -1 INDICEMAX 9
   1          (potencia más pequeña)
   BEGIN
        DUP,(incluir potencia)
        1 INDICEMAX+!
        3 UM* (siguiente potencia, doble longitud)
                  (reemplazar UM* por U* en FORTH-79 y
                  figFORTH)
   UNTIL          (hasta que la siguiente potencia lleve
                  más de 2 bytes)
   DROP           (bytes menos significativos de potencia
                  de doble longitud)
;
CREATE 3POTENCIAS POTENCIAS,
                  (no es necesaria ALLOT., realiza la
                  inclusión)
:3**              (n--3**n)
   INDICEMAX @ OVER U < IF
```

En estrecha formación

Definir una estructura de semimatriz no es difícil gracias a CREATE y DOES>. El diagrama refleja la respuesta de FORTH a la entrada 11 1MATRIZ EJECUTA, donde 1MATRIZ se ha incluido previamente en el diccionario como una def. de dos puntos. También refleja el efecto sobre la pila

Def. de dos puntos	11 1MATRIZ EJECUTA	Pila
:1MATRIZ CREATE	CREATE toma EJECUTA, le asigna un c. de nombre y un c. de código. Cuando se utilice EJECUTA, dejará su pfa en la pila	11
DUP +	DUP + multiplica 11 por dos y pone el resultado en la pila	22
HERE	HERE coloca en la pila la siguiente dirección disponible en el diccionario. Puesto que acabamos de incluir EJECUTA, ésta es la pfa de EJECUTA	PFA 22
OVER	OVER copia 22 en la parte superior de la pila, encima de la pfa de EJECUTA	22 PFA 22
ALLOT	ALLOT quita 22 de la pila e incluye 22 bytes en el diccionario, que (dado que acabamos de incluir EJECUTA) constituyen el campo de parámetros para EJECUTA	PFA 22
SWAP 0 FILL	SWAP pone 22 encima de la pfa de EJECUTA; coloca 0 en la pila; luego FILL toma los tres parámetros y pone a 0 los 22 bytes del c. de parámetros	0 22 PFA → VACÍA
DOES>	RUN-TIME (después de DOES>) 99 2 EJECUTA! (p. ej.) prepara la pila como vemos y luego almacena el valor 99 en el elemento 2 de la matriz. Se EJECUTA del siguiente modo:	PFA 2 99
SWAP	SWAP coloca en la parte superior de la pila el "subíndice de matriz" (2)...	2 PFA 99
DUP +	DUP + lo duplica...	4 PFA 99
+ ;	+ le suma el subíndice (que, efectivamente, está actuando como un valor decalado) a la pfa de EJECUTA...	PFA +OFF-SET
!	! toma 99 y lo almacena en la dirección correcta	VACÍA

Caroline Clayton

```
        DROP  0 (n no entre 0 e INDICEMAX.
              Dejar resultado 0)
    ELSE          (querer contenido de memoria
                  en 3POTENCIAS+2*n)
        2*3POTENCIAS + @
    THEN
;
```

3POTENCIAS era una matriz numérica unidimensional. Usted podría igualmente haberla preparado utilizando ALLOT (recuerde que tiene que asignar dos bytes para cada elemento de la matriz) y !. Las matrices de caracteres son similares, pero puesto que cada carácter representa sólo un byte, usted debe utilizar C,, C! y C@ en lugar de ,, ! y @.

La estructura CREATE

Siempre que emplee una matriz como 3POTENCIAS, necesita hacer algo como:

```
2*3POTENCIAS+
```

Ello se debe a que 3POTENCIAS no sabe que es una matriz. Simplemente deja la dirección de su campo de parámetros y deja que usted escoja qué hacer con ella. Para hacer palabras más "inteligentes", hay una construcción muy interesante que emplea CREATE (o<BUILDS en figFORTH) y DOES>. Éstas permiten definir palabras nuevas que sean versiones mejoradas de CREATE, de modo que en realidad son nuevas palabras definitorias. Ellas sí crean (CREATE), pero usted también puede decirles que hagan otras cosas al mismo tiempo que el CREATE. Asimismo, pueden decirle a usted qué ha de hacer la palabra recién creada cuando la utilice (en vez de limitarse a apilar la dirección de su campo de parámetros).

He aquí un ejemplo que crea matrices inteligentes:

```
:1MATRIZ  (n--)
     (crea una matriz numérica unidimensional
     con dimensión n)
   CREATE       (el nombre de copia de 1MATRIZ
                cuando usted la usa)
   DUP+         (una forma rápida de hacer 2*)
   HERE         (recuerde la dirección del campo de
                parámetros para FILL)

   OVER ALLOT   (ahora tiene 2*,pfa)

   SWAP 0 FILL  (poner a cero todos los bytes del
                campo de parámetros)

DOES>           (subíndice, pfa -- dirección de
                elemento)
   SWAP DUP + (pfa,2*subíndice)
```

Se divide en dos partes: la parte de tiempo de definición, antes de DOES>, se lleva a cabo cuando usted utiliza 1MATRIZ y prepara el campo de parámetros de la palabra siguiente.

Supongamos que dice:

```
11 1MATRIZ EJECUTA
```

Ésta crea una nueva palabra, EJECUTA, y le destina (ALLOT) un campo de parámetros de 22 bytes inicializados en cero. La segunda parte de 1MATRIZ, después del DOES>, es la parte de tiempo de ejecución. Ésta se realiza cuando se utiliza EJECUTA. EJECUTA coloca la dirección de su campo de parámetros en la pila, pero después de que la parte de tiempo de ejecución de 1MATRIZ aplique su inteligencia.

En efecto, ahora usted tiene 11 variables, denominadas:

```
 0 EJECUTA
 1 EJECUTA
 :
 :
10 EJECUTA
```

Las emplea tal como si fueran variables comunes, p. ej.:

```
99 2 EJECUTA !
4 EJECUTA @.
```

Otras aplicaciones

Ésta era una aplicación bastante directa de CREATE...DOES>. Pero una vez que aprenda a usarla, descubrirá que puede hacer cosas mucho más inteligentes. Por ejemplo, si ha comenzado a escribir sus propios programas en FORTH, sin duda en muchas ocasiones se habrá olvidado de incluir el @ al definir variables. Con CREATE y DOES> usted puede definir una clase de variable que no emplea el signo @.

Normalmente, sólo deja en la pila su valor, como una constante, pero surge un problema cuando usted desea cambiar su valor. Es mejor, entonces, el uso de una palabra, ->, como en:

```
nuevo valor -> nombre del nuevo estilo de
   variable
```

-> no hace nada, exceptuando el hecho de que deja una "nota" (estableciendo una variable) diciendo que se la ha utilizado, y ésta le dice a la variable de "nuevo estilo" que, en vez de apilar su valor actual, debe tomar un nuevo valor de la pila.

```
VARIABLE -> USADA
0 -> USADA !
: -> -1-> USADA!;
```

Ahora deseamos un recambio para la palabra definitoria VARIABLE.

Esta palabra definitoria (llamémosla VAR) constituye una nueva palabra de este tipo, utilizando CREATE...DOES>:

```
:VAR(--)
   CREATE
   0,(destinar 2 bytes, inicializados en 0)
DOES>
   —>USADA @ IF (nuevo valor, pfa--)
      !          (establecer nuevo valor)
      0 —> USADA !
   ELSE        (pfa--valor actual)
      @
   THEN
;
```

Veamos cómo definir una variable VAR x, establecerla en 99 y luego imprimir su valor:

```
VARx
99—> x
x.
```

La razón por la cual CREATE y DOES> son tan útiles es que permiten que usted combine dos propieda-



des que suelen ir separadas. La mayoría de los lenguajes de programación poseen datos, que son "pasivos", y requieren que se actúe sobre ellos, y rutinas, que son "activas" pero necesitan recibir algunos datos.

Con CREATE y DOES> usted puede establecer datos inteligentes que actuarán sobre sí mismos utilizando una parte DOES> de tiempo de ejecución. Esta idea es muy fructífera; pero para poder valerse de ella debe olvidarse de la distinción entre datos y rutinas en la que hacen tanto hincapié casi todos los otros lenguajes.

Palabras útiles

,(n– –) Incluye n como dos bytes en el diccionario. Hay una versión C, de un byte.
HERE(– – dirección) Apila la dirección del lugar del diccionario donde se llevará a efecto la siguiente inclusión.
FULL(dirección,n,relleno– –) Rellena n bytes, comenzando en la dirección dada, con el byte de relleno dado.

Matriz de matrices

Las matrices definidas por 1MATRIZ sólo son "levemente" inteligentes. Trabajan con bastante rapidez, pero no comprueban que el subíndice que se les da sea razonable, y existe el riesgo de que accidentalmente usted escriba sobre algo fuera de la propia matriz. Si 1MATRIZ almacenara la dimensión al comienzo del campo de parámetros, la parte en tiempo de ejecución podría comprobar que el subíndice proporcionado sea menor que la dimensión, y denunciarlo si no lo fuera. Algunos lenguajes, como el PASCAL y el BASIC, comprueban los subíndices de matrices como ésta, mientras que otros, como el C, dan prioridad a la velocidad y no efectúan comprobaciones. El FORTH obliga al usuario a calcular lo que realmente desea y también a hacerlo en la forma debida. Los lenguajes también difieren en el uso o no del subíndice 0 para el primer elemento o del subíndice 1, o en realidad de cualquier otro subíndice. Nuevamente, en FORTH usted puede elaborar el contenido de las matrices a la medida de su propio gusto. Las matrices multidimensionales son similares en principio, pero más complicadas. Normalmente uno se imagina a la matriz multidimensional como un bloque rectangular o tridimensional (o peor), pero el ordenador la almacena fila por fila en una larga lista. Usted convierte sus diversos subíndices en un decalaje en esta lista mediante un proceso consistente en multiplicar por las dimensiones.

Por ejemplo, para una matriz tridimensional de dimensiones 2, 3 y 4 (24 elementos en total), usted convierte los subíndices i, j y k en un:

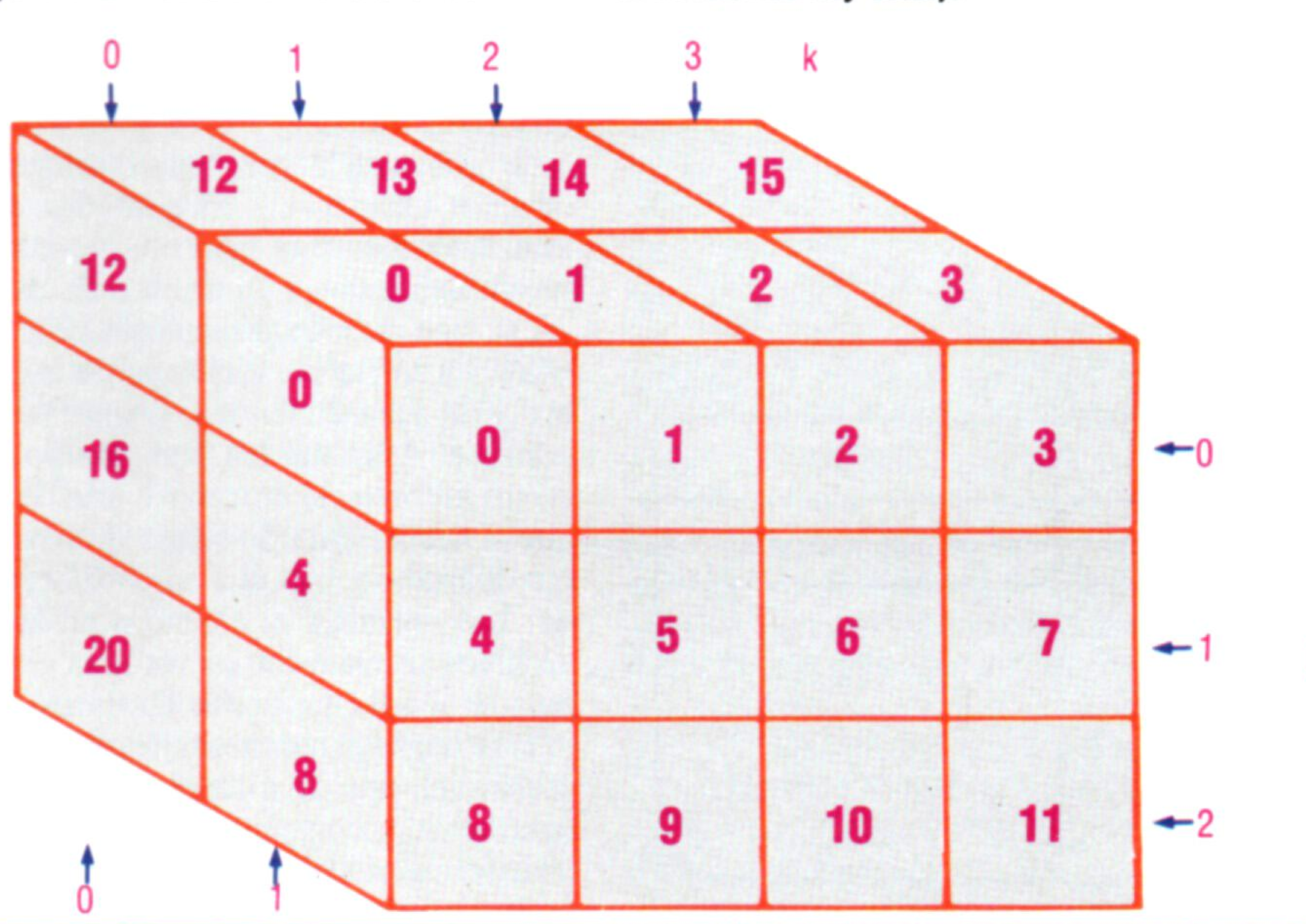
(i*3 + j)*4 + k

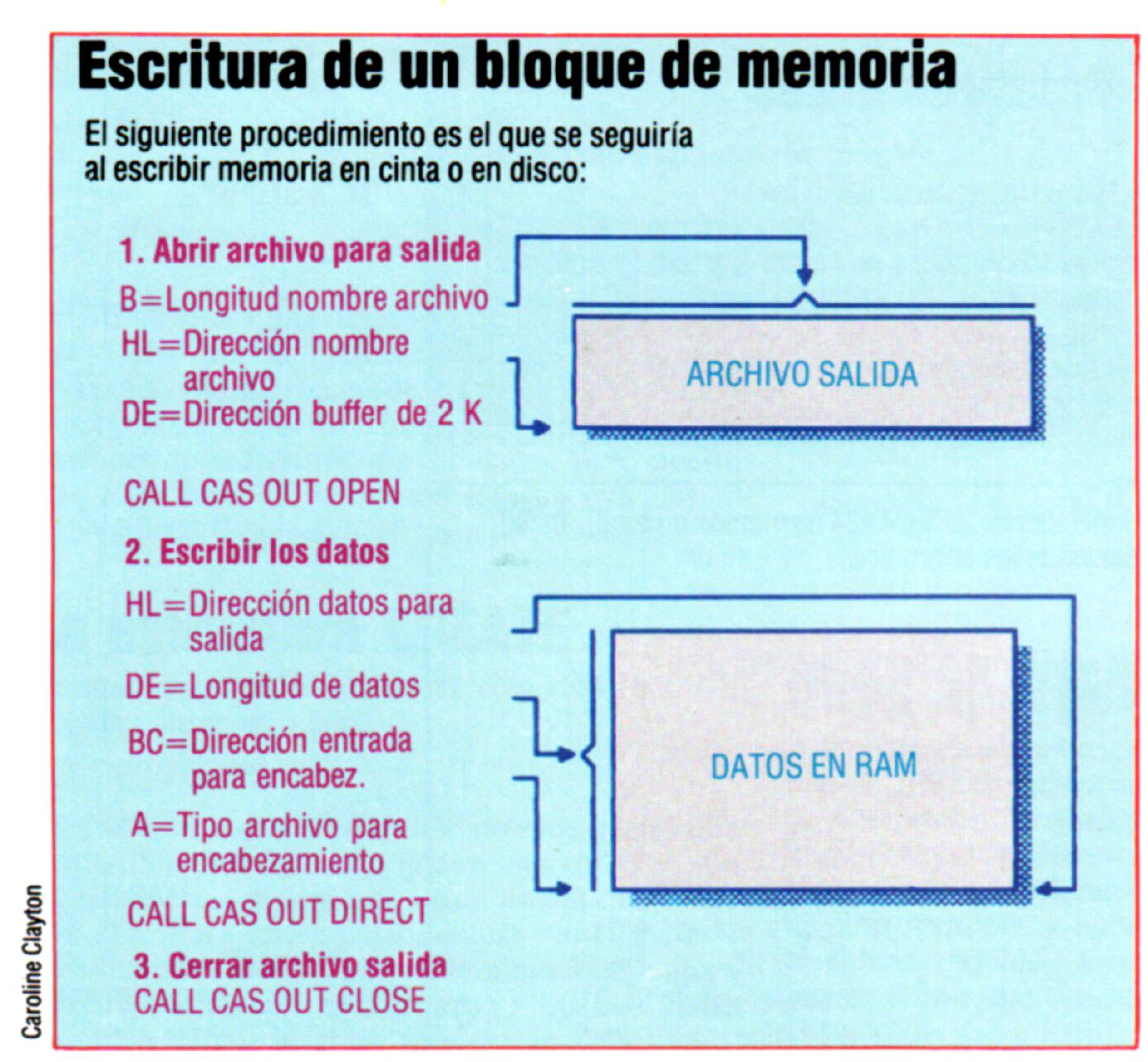
He aquí cómo definiría 3MATRIZ:

```
:3MATRIZ (d,e,f– –)
   (crea una matriz tridimensional con
   dimensiones d, e y f)
CREATE
   OVER,DUP,          (incluir e y f)
   HERE               (d,e,f,pfa + 4)
   2ROT*ROT*ROT*      (pfa + 4,2* d* e* f)
   DUP ALLOT          (incluir 2* d* e* f bytes)
   0 FILL             (inicializarlos en 0)
DOES>                 (i,j,k,pfa– –dirección)
   DUP@               (i,j,k,pfa,e)
   4ROLL*             (j,k,pfa,i*e)
   3ROLL+             (k,pfa,i*e + j)
   OVER2+@            (k,pfa,i*e + j,f)
   *                  (k,pfa,[i* e + j]* f)
   ROT + DUP +        (pfa,2*[[i* e + j]* f + k])
   SWAP 4 ++
;
```

(Observe que en el FORTH-79 el número de antes de ROLL debe ser uno mayor.)

Se pueden escribir palabras análogas (4MATRIZ, 5MATRIZ, etc.) a lo largo de líneas similares. Si esto le parece complicado, la mejor forma de comprenderlo es ver cómo lo hace el C. Incluso se puede escribir una palabra definitoria, NMATRIZ, que tome un número de la pila diciéndole cuántas dimensiones hay debajo





Caroline Clayton

Formato del encabez.

DIRECCIÓN	CONTENIDO
0 – 15	Nombre archivo
16	Número bloque
17	Flag ult. bloque
18	Tipo archivo
19 - 20	Longitud datos
21 - 22	Posición datos
23	Primer bloque
24 - 25	Longitud lógica
26 - 27	Dirección entrada
28 – 63	No definido

La apertura de un archivo para su carga mediante CAS IN OPEN hace que el encabezamiento del archivo se escriba en la RAM y su dirección quede en HL. Además, DE contendrá la posición original de los datos (como indica el encabezamiento), BC contendrá la longitud lógica del archivo, y A el tipo de archivo. La información retenida en el byte del tipo de archivo es de bits significativos, como se muestra aquí

El tipo de archivo se define como varios campos diferentes:

Bit 0	Bits 1-3	Bits 4-7
Protección	Contenido archivo	Versión
0=Protegido 1=No protegido	0=BASIC 2=Binario 4=Pantalla 6=ASCII 8=Sin colocar	16=ASCII 32=Sin colocar

Caroline Clayton

A buen recaudo

Tanto la platina de cassette del Amstrad CPC 464 como la unidad de disco del CPC 664 aseguran un almacenamiento eficaz y fiable

El Amstrad CPC 664 ha dado un paso adelante al sustituir la platina de cassette del CPC 464 por una unidad de disco, con lo que ha incrementado aún más la rapidez de acceso. Naturalmente, el equipo físico no serviría para mucho si detrás no estuviera el adecuado software que lo soportara. La pregunta es: ¿qué se ha previsto en el firmware para que el programador en código máquina emplee estas facilidades de almacenamiento?

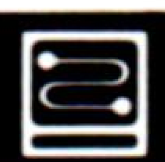
Un análisis de las características del firmware podría dividirse en tres áreas: las que son generales a la transferencia de archivo, las que son exclusivas del sistema de cinta y las que son exclusivas de la versión de disco.

Por lo general, cada archivo contiene dos secciones independientes de información: la primera es la sección de encabezamiento y la segunda es la de los datos. El encabezamiento contiene todas las informaciones que necesita el firmware para determinar cómo debe tratarse el archivo (por ejemplo, el tipo de archivo, dónde debe cargarse y su longitud). Los datos son simplemente el contenido del archivo. El firmware proporciona al usuario toda la información acerca del archivo y la facilidad encargada de manejarlo en memoria.

La sección del firmware que nos interesa es el gestor de cassette, aunque este término es algo impreciso, dado que también se emplean los mismos elementos para manipular los archivos de disco.

El gestor de cassette emplea dos "corrientes", una para entrada y otra para salida, que pueden estar activas a la vez. Esta configuración no resulta de gran utilidad en un grabador de cinta, pero es una facilidad muy potente cuando se emplea correctamente con una unidad de disco. Las corrientes se han de inicializar antes de poder accederse a ellas. La inicialización consiste en asignar un nombre y un área de trabajo (*buffer*) a la corriente que interesa. A un mismo tiempo sólo puede estar activo un archivo de entrada y uno de salida, de modo que si la corriente de salida no se emplea, puede no ser definida como una segunda corriente de entrada. Las entradas interesadas en la transferencia de archivos se encuentran listadas en la tabla de rutinas de almacenamiento de datos.

Los datos pueden ser transferidos a o desde el archivo en uno de estos dos modos: *transferencia de caracteres* o *transferencia de bloques*. En el modo transferencia de caracteres, el firmware utiliza el *buffer* asignado a la corriente para almacenar los



datos temporalmente. Esto significa que se puede efectuar el acceso tanto hacia como desde una cinta (o un disco) en grandes bloques mientras el programa trata los datos de byte en byte. Naturalmente, un programa que tenga que acceder al disco cada vez que se necesita un carácter sería muy lento. Los dos modos de transferencia se excluyen: una vez efectuado el acceso a un archivo en un modo es imposible cambiar y acceder al mismo en el otro.

En el modo de transferencia por bloques (CAS IN DIRECT y CAS OUT DIRECT) los *buffers* asignados a las corrientes no se emplean: en su lugar, los datos se leen directamente desde o hacia la memoria. La memoria puede estar en la RAM o en la ROM.

Las rutinas CAS IN CLOSE y CAS OUT CLOSE son especialmente importantes porque informan al firmware que la corriente está cerrada y, por ende, disponible para un nuevo archivo. Si CAS OUT CLOSE no se efectúa después de una salida de carácter, el *buffer* de salida puede quedar incompleto y, por consiguiente, no escrito del todo. CAS OUT CLOSE evita este problema obligando al firmware a enviar cualquier bloque incompleto de datos.

Nuestro esquema muestra el proceso por el cual un bloque de memoria se escribe en un archivo. Ante todo, el archivo es abierto por medio de CAS OUT OPEN, que se limita a asignar un nombre al fichero y reserva la corriente de salida para el empleo del archivo. Los datos son escritos desde la memoria empleando CAS OUT DIRECT, y finalmente el archivo se completa llamando a CAS OUT CLOSE que, en este caso, sólo indica que la corriente de salida está disponible para su empleo.

El gestor de cassette emplea un área de datos, conocida como el *encabezamiento*, para almacenar información correspondiente a cada archivo individual. En el caso de una operación de lectura, esta información es tomada desde el comienzo del archivo y, en el caso de una escritura, el encabezamiento se establece en memoria y después se escribe al comienzo del archivo. El planteamiento general del encabezamiento se muestra en el diagrama *Formato del encabezamiento*.

A nivel de firmware, el tipo de archivo no tiene efecto alguno: el archivo puede ser leído en ambos sentidos independientemente del tipo. El tipo resulta importante cuando el firmware es empleado por un programa de alto nivel, p. ej., en BASIC.

La *posición de datos* se define como la posición desde la cual se escribió originalmente el archivo. Se observará que esto sólo es importante cuando el archivo es escrito directamente desde la memoria. Cuando se emplea una salida de carácter, la posición representa la dirección del *buffer* de salida. La

Rutinas de almac. de datos

Sistema cassette:

$BC65	CAS INITIALISE	Esta rutina asegura el establecimiento de las condiciones por omisión: es decir, las corrientes de entrada/salida están cerradas y los mensajes habilitados
$BC6B	CAS NOISY	Habilita y deshabilita los mensajes proporcionados desde el gestor de cassette
$BC77	CAS IN OPEN	Inicializa un archivo para leer desde él
$BC80	CAS IN CHAR	Lee un único carácter del archivo entrada
$BC83	CAS IN DIRECT	Carga en memoria un archivo entero
$BC89	CAS TEST EOF	Comprueba el final de archivo entrada
$BC7A	CAS IN CLOSE	Cierra el archivo entrada actual
$BC7D	CAS IN ABANDON	Cierra el archivo entrada, aún sin llenar del todo
$BC8C	CAS OUT OPEN	Inicializa un archivo para escribir en él
$BC95	CAS OUT CHAR	Envía un solo carácter al archivo salida
$BC98	CAS OUT DIRECT	Escribe un bloque de memoria en el archivo salida
$BC8F	CAS OUT CLOSE	Cierra un archivo ya completo
$BC92	CAS OUT ABANDON	Cierra el archivo descartando la información no escrita
$BC9B	CAS CATALOG	Genera un catálogo de archivos

Sistema de disco:

$BE80	SET MESSAGE	Habilita y deshabilita mensajes dados por el disco
$BE83	SETUP DISK	Inicializa los tiempos de la unidad (velocidad de pasos, retardo carga/descarga y tiempo agotado *on/off*)
$BE86	SELECT FORMAT	Establece el formato del disco
$BE89	READ SECTOR	Lee un sector del disco
$BE8C	WRITE SECTOR	Escribe un sector del disco
$BE8F	FORMAT TRACK	Formatea una pista
$BE92	MOVE TRACK	Mueve el cabezal de la unidad hasta una pista determinada
$BE95	GET DR STATUS	Proporciona un byte que indica el estado de la unidad actual
$BE98	SET RETRY COUNT	Establece el número de retiradas en caso de error

Las operaciones de cassette o disco se efectúan por medio de llamadas del firmware. Algunos puntos que hay que anotar sobre el sistema de archivado del Amstrad son la facilidad para mantener abiertos tanto los archivos de entrada como los de salida simultáneamente y la casi completa compatibilidad de la síntaxis de manejo de los archivos en cassette y en disco. Los archivos pueden ser cargados o grabados (*read out, read in*) directamente de la RAM (empleando CAS IN/OUT DIRECT) o carácter a carácter a través de un *buffer* de 2 Kbytes (CAS IN/OUT CHAR). No obstante, una vez determinado, el modo de entrada/salida no puede cambiarse

longitud lógica es el número total de bytes de datos en el archivo. Obsérvese que ésta no incluye encabezamiento de información, o cualquier "almohadilla" que se haya requerido para llenar el *buffer*.

La *dirección de entrada* es la dirección en la que se inician los programas en código máquina cuando ocurre un *auto-run* desde el BASIC. Los otros campos son específicos sólo para tratamiento de archivos en cassette y, por ende, su uso es muy limitado.

La detección de errores es efectuada automáticamente por el firmware por medio de un código CRC, aunque son raros los errores que se encuentran merced a la meditada programación del sistema. Los errores que aparezcan son señalados por el firmware para que el usuario decida qué hacer.

A fin de facilitar el tratamiento de errores de lectura, los archivos en cassette no se escriben en su integridad, sino divididos en bloques de dos Kbytes. Cada bloque es tratado como una unidad distinta, y por ello se puede volver a cargar un bloque "malo" sin tener que recargar todos los bloques restantes que le precedieron. Este planteamiento por bloques retarda necesariamente la velocidad de transferencia del archivo, pero la eficacia que se consigue compensa la pérdida de tiempo. La velocidad de escritura a la cassette puede variar bajo control del software por medio de la entrada CAS SET SPEED en $BC68, mientras que la velocidad de lectura es seleccionada automáticamente por el firmware.

Una de las configuraciones más útiles del sistema operativo del Amstrad, en lo que respecta al programador, es la instalación transparente de instrucciones de disco. Es decir, todas las rutinas generales de archivo tienen exactamente los mismos parámetros y, por tanto, un programa escrito para operaciones en cassette puede trabajar automáticamente con la unidad de disco sin modificación alguna. Esto se ha logrado gracias a la técnica del *parcheo*, que ya describimos anteriormente. Sin embargo, esta técnica no es inmediata, por lo que unas cuantas sugerencias harán que su empleo no resulte tan problemático.

Las áreas particulares que hay que vigilar son las que se refieren al empleo de memoria. El sistema de disco es controlado desde una ROM de ampliación, aunque también necesita una cierta cantidad de RAM de espacio de trabajo. Ésta se toma generalmente de la parte superior de la memoria (pero puede reasignarse a otras áreas de la RAM). Por tanto, cualquier programa que quiera usar el disco debe tener en cuenta que esta área de RAM ($504H bytes por debajo de HIMEM) no está disponible.

Otro aspecto, derivado de la implementación de la compatibilidad del nombre de archivo de CP/M, es que los nombres de archivo en disco no pueden contener más de 12 caracteres. Por esto es mejor suponer una longitud de ocho caracteres para evitar así problemas de traslación de cinta a disco.

Los encabezamientos de disco se comportan de una manera algo diferente de los encabezamientos de cinta; las referencias a los bloques aquí ya no tienen importancia y son ignoradas. El encabezamiento es también sometido a una suma de chequeo, para que el firmware pueda distinguir entre archivos CP/M y AMSDOS.

La adición del sistema de disco conlleva entradas adicionales del bloque de saltos, que están a disposición del usuario para el manejo de discos a nivel de sistema operativo. Esto es lo que se detalla en la segunda parte de nuestra tabla.

Por último, damos aquí un programa de ejemplo que ilustra cómo puede extraerse la información de un encabezamiento de archivo. El archivo primero se abre por medio de una llamada a CAS IN OPEN, que genera el encabezamiento en la RAM. Una vez extraída la información pertinente, el archivo se descarta mediante CAS IN ABANDON antes de que tenga lugar cualquier lectura. El programa en BASIC asociado sirve para proporcionar una interface de usuario a la rutina.

```
10 'Lector encabezamiento archivo
20 'Emplea CAS IN OPEN en BC77H
30 routine=&8800: address=&8800: ON ERROR GOTO
   530
40 MODE 1: MEMORY &5FFF
50 WHILE -1
60 CLS
70 LOCATE 10,10: PRINT "¿Archivo cassette o disco?"
   ;CHR$(18)
80 a$=""
90 WHILE a$<>"T" AND a$<>"D"
100 a$=UPPER$(INKEY$)
110 WEND
120 IF a$="T" THEN ITAPE ELSE IF a$="D" THEN IDISC
130 LOCATE 10,12: INPUT "Nombre del archivo"
140 IF LEN(file$)=0 AND a$="D" THEN 130
150 GOSUB 420
160 CALL routine
170 buffer=PEEK(address)+256*PEEK(address+1)
180 LOCATE 20,12: ff=0
190 FOR name=0 TO 7-8*(t$="T")
200 char=PEEK(buffer+name)
210 IF (char<32 OR char>126) AND char>0 THEN ff=1:
    GO TO 230
220 PRINT CHR$(char);
230 NEXT
240 IF ff=1 THEN PRINT CHR$(8); "No hay nombre
    archivo";
250 PRINT
260 'test filetype
270 LOCATE 5,16: PRINT "Tipo archivo: ": LOCATE 20,16
280 type=PEEK(buffer+18)
290 IF (type AND 1)=1 THEN PRINT "Protegido";
300 file = type AND &FE
310 IF file=0 THEN PRINT "Basic"
320 IF file=2 THEN PRINT "Binario"
330 IF file=6 THEN PRINT "ASCII"
340 IF file=6 THEN PRINT "Desconocido"
350 length=PEEK(buffer+24)+256*PEEK(buffer+25)
360 LOCATE 5,18: PRINT "Longitud datos:" :LOCATE
    20,18: PRINT HEX$(length,4)
370 entry=PEEK(buffer+26)+256*PEEK(buffer+27)
380 LOCATE 5,20: PRINT "Direccion entrada:" :LOCATE
    20,20: PRINT HEX$(entry,4)
390 LOCATE 9,25:PRINT "Pulsar una tecla para continuar"
400 WHILE INKEY$=" ":WEND
410 WEND
420 RESTORE 520: READ byte: off=0
430 WHILE byte<>999
440 POKE routine+off, byte
450 off=off+1: READ byte
460 WEND
470 POKE routine+1,LEN(file$)
480 FOR char=1 TO LEN(file$)
490 POKE routine+&100+char-1,ASC(MID$(file$,char,1)).
500 NEXT
510 RETURN
520 DATA &06,&00,&21,&00,&89,&11,&00,&60,
    &cd,&77,&bc,&7d,&32,&0,&80,&7c,&32,&01,&80,
    &cd,&7d,&bc,&c9,999
530 'trampa de errores
540 IF ERL=80 THEN a$="":RESTORE
550 STOP
```

mi computer

Editorial Delta, S.A.



10102

9 788485 822836